

Popularized by the Star Wars saga, complex space planes which can move between planet surfaces and space with ease, are now the staple diet of gamers worldwide. The maths is extreme, which is why we take care of this for you.

NASA would call this a Space Plane controller. It can control a "craft" like an airplane while inside a planet's atmosphere, but can also act like a spaceship while in space.

What is Sci-Fi Ship Controller?

Sci-Fi Ship Controller is an asset that allows you to quickly and easily turn your ship models into fully-functioning, flying ships.

How do I use it in Unity?

Currently, it consists of three main modules: The Ship Control Module, the Player Input Module, and the Ship AI Input Module. The Ship Control Module is a script that can be added to ship models to turn them into flyable ships, complete with all the parameters needed to tweak their behaviour to your liking. The Player Input Module is a script that can be added to any ship with the Ship Control Module already attached to map inputs from a wide variety of input sources to the Ship Control Module in order to let a player control the ship.

Why should I use Sci-Fi Ship Controller instead of another asset?

One of the main things we've worked really hard on with Sci-Fi Ship Controller is its ease of use. All the parameters in the modules are arranged logically and have headers describing their functionality. In addition to this, every editable parameter has an associated tooltip, so if you're unsure about what something does you can simply hover your mouse over it and get a brief description. We've also tried to write Sci-Fi Ship Controller in a way that makes sense to all game developers, not just ones that have a degree in aerodynamic engineering or physics.

One of the other strengths of Sci-Fi Ship Controller is its versatility. It isn't an asset JUST for arcade spaceships, JUST for aircraft or JUST for hover-ships: If it flies, then there's a good chance you can make it with our asset (and if you can't, feel free to let us know so that we have the opportunity to improve Sci-Fi Ship Controller in that regard).

Sci-Fi Ship Controller includes an extensive, documented, runtime API with many C# examples.

As well as this, Sci-Fi Ship Controller is completely physics-based. All movements are driven by Unity's built-in physics, which provides a great feel for players and ensures you won't encounter any strange behaviour caused by our asset fighting with the physics engine.

Finally, Sci-Fi Ship Controller is designed from the ground up for performance. We've tried as much as possible to follow best practice and avoid expensive allocations and function calls, so Sci-Fi Ship Controller should only comprise a minimal part of your performance budget. The Combat system supports DOTs (Entities, Jobs, Burst Compiler in Unity 2020.3).

What versions of Unity do we support?

We follow a sliding window which roughly matches the versions supported by Unity. We currently support Unity 2020.3.25+, 2021.x, 2022.x, and 2023.1. We recommend using LTS versions.

Table of Contents

What's Changed	8
Support Policy	9
Getting Started.....	10
Videos and Tutorials.....	10
Demos	11
General Demos.....	11
AG SSC Racer	11
Tech Demo 2	11
Tech Demo 3	12
Tech Demo 4	13
Ship Control Module	13
Ship Control Module – Overview	13
In-Scene Editing.....	13
Physics Tab.....	14
Control Tab.....	15
Thrusters Tab	18
Aero Tab.....	20
Combat Tab.....	22
Player Input Module	29
Direct Keyboard	29
Legacy Unity.....	30
Rewired	30
Unity Input System.....	32
Virtual Reality (VR) Input	33
Unity XR - Overview	34
Unity XR – Levers and Joysticks with Sticky3D.....	36
Unity XR – Hands and Physics Collisions	36

Custom Player Input.....	37
Using Sony Dual Shock 4 with SSC on PC	38
Overriding Player Input Module in Code	38
Writing your own Player Input code	38
Ship Camera Module.....	38
Camera Overview	38
Camera Properties	38
Top-down Setup.....	41
Ship Camera Settings ScriptableObject.....	41
Projectile Module.....	41
How to Create Projectile Prefabs.....	41
How to Setup Projectiles to use DOTS/ECS.....	42
How to Convert scene to use DOTS projectiles	43
Projectile Properties	44
How to Use Guided Projectiles	45
Customising Projectile Behaviour	45
Beam Module.....	45
How to Create Beam Prefabs.....	45
Beam Properties.....	46
Destruct Module	46
How to Create Destruct Prefabs	47
Destruct Properties.....	47
Effects Module	48
SSC Manager	49
Adding SSC Manager to a Scene	49
Locations and Paths - Overview.....	49
Locations and Paths - Creating.....	49
Editing Paths	50
Location Properties.....	50
Path Properties	51
Object Pool Estimation.....	51
Ship AI System.....	52
Ship AI Overview	52
Ship AI Input Module	52
AI State Interaction	53
Setting a Ship's AI State	53
Getting a Ship's AI State.....	53

Setting State Inputs.....	54
Getting State Inputs.....	55
State Completion Status	56
State Stage Index	56
Sample AI Scripts.....	57
Default AI States.....	57
Custom AI States	58
Creating and Using a Custom State.....	58
Writing a Custom State Method	58
State Behaviour Combiner	60
Default AI Behaviours	61
SSC Radar	62
General Properties	62
Visual Properties	62
Movement Properties.....	63
Surface Turret Module.....	63
Surface Turret - General Properties.....	63
Surface Turret - Weapon Settings.....	64
Surface Turret - Gravitational Properties.....	65
Surface Turret – Optional Components	65
Auto Targeting Module.....	65
Damage Receiver	66
Destructible Object Module.....	66
Ship Docking and Undocking.....	67
Ship Docking Station - Overview	67
Ship Docking Station - General Properties.....	68
Ship Docking Station – Docking Point Properties	69
Ship Docking Component.....	70
Player Docking – Non-assisted.....	73
Player AI-Assisted Docking.....	73
Ship Display Module (HUD).....	73
Ship Display Module – Extending.....	74
Ship Display Module – General Settings	74
Ship Display Module – Display Reticle Settings	75
Ship Display Module – Altitude and Speed Settings	76
Ship Display Module – Display Attitude Settings	76
Ship Display Module – Display Fade Settings.....	77

Ship Display Module – Display Flicker Settings	77
Ship Display Module – Display Heading Settings	78
Ship Display Module – Display Message Settings	78
Ship Display Module – Display Target Settings	79
Ship Display Module – Display Gauge Settings	80
Ship Warp Module	82
Warp Overview	82
Warp General Tab	82
Warp Ship Tab	83
Warp Camera Tab	83
Warp FX Tab	84
Warp Events Tab	84
Scriptable Render Pipelines	85
Proximity Component	85
Ship Proximity Component	85
SSC Moving Platform	86
SSC and Unity Physics (DOTS)	87
Common Issues	88
Common Issues – General	88
Common Issues – Auto Targeting Module	88
Common Issues – Player Input	89
Common Issues – Ship Behaviour	89
Common Issues – Ship AI Behaviour	90
Common Issues – Ship Camera Module	90
Common Issues – Destruct Module	90
Common Issues – Demo Scenes	90
Common Issues – Demo Prefabs and Scripts	90
Common Issues – Effects	90
Common Issues – Radar	91
Common Issues – Weapons	91
Common Issues – Moving Platforms	91
Common Issues – Paths	92
Common Issues – Ship Docking Station	92
Common Issues – Ship Display Module (HUD)	92
Common Issues – Ship Warp Module	93
Common Issues – SSC Celestials	93
Common Issues – VR	93

Runtime and API	94
Runtime General Guidance	94
Changing Variable at Runtime	95
Demo Scripts	95
Ship Control Module Methods or Properties	99
Ship Control Module API Call Backs	102
Ship (General) Methods or Properties	103
Ship (Damage) Methods or Properties	105
Ship (Respawning) Methods or Properties	107
Ship (Thruster) Methods or Properties	107
Ship (Weapon) Methods or Properties	108
Ship API Call Backs	111
Player Input Module Properties	111
Player Input Module (General) API Methods	112
Player Input Module (AI-Assist) API Methods	113
Player Input Module (XR) API Methods or Properties	114
Ship AI Input Module API Methods or Properties	115
Ship AI Input Module API Call Backs	118
Ship Camera Module API Methods	119
Auto Targeting Module API Methods or Properties	120
Auto Targeting Module API Call Backs	121
Location and Path API Methods	122
Beam, Destruct, Projectile and Effects API Methods	124
Projectile API Call Backs	126
Destructible Object Module API Methods or Properties	126
Radar API Methods or Properties	128
Radar API Call Backs	132
Surface Turret Module API Methods	132
Surface Turret Module API Call Backs	133
Ship Docking Station API Methods	133
Ship Docking API Methods	136
Ship Docking API Call Backs	137
Ship Display Module API Properties	137
Ship Display Module (General) API Methods	137
Ship Display Module (Cursor) API Methods	138
Ship Display Module (Display Reticle) API Methods	138
Ship Display Module (Altitude and Speed) API Methods	139

Ship Display Module (Display Attitude) API Methods	140
Ship Display Module (Display Fade) API Methods	140
Ship Display Module (Display Flicker) API Methods	140
Ship Display Module (Display Gauge) API Methods	141
Ship Display Module (Display Heading) API Methods.....	143
Ship Display Module (Display Message) API Methods.....	144
Ship Display Module (Display Target) API Methods	146
Ship Display Module API Call Backs	148
Ship Warp Module (General) API Properties	149
Ship Warp Module (Camera) API Properties	149
Ship Warp Module (General) API Methods	149
Ship Warp Module (Ship) API Methods	150
Ship Warp Module (Camera) API Methods.....	150
Ship Warp Module (FX) API Methods	150
Ship Warp Module (Events) API Methods	151
Support.....	151
Version History.....	151
Trade Marks	160

What's Changed

Version 1.4.1

- [NEW] Ship Docking - snap position and rotation axes options
- [NEW] ShipControlModule - callbackOnThrusterSystemsStatusChanged
- [NEW] ShipControlModule - thruster list copy and paste buttons
- [NEW] AutoTargetingModule - onTargeted callback
- [FIXED] Thrusters may fire when Thruster Systems are offline
- [FIXED] Thrusters may pause when ship completes docking
- [FIXED] Thruster may fire when Min FX Always On and throttle = 0
- [IMPROVED] Updated SRP packages to URP and HDRP 10.7.0

Version 1.4.0

- [NEW] Ship Display Module – Fade in or out options with API
- [NEW] DemoSSCSequenceRenderers component
- [NEW] Ship Proximity component
- [NEW] Ship AI Input Module - ability to enable or disable main update loop
- [FIXED] SSCManager may not be created in correct scene when using additive scenes
- [FIXED] Copied style is null. Using StyleNotFound instead (U2022.3 only)
- [FIXED] ShipDocking - on undock event does not fire if there is no exit path
- [IMPROVED] Ship Camera Module - Check Update Type if Lock To Pos or Rot

Version 1.3.9

- [NEW] AddWeapon API
- [FIXED] May not respawn at correct location in U2022+ with Vsync or targetFrameRate set
- [IMPROVED] Sticky3D Controller first-person integration with TechDemo3
- [IMPROVED] Compatibility with Unity 2022.3 and 2023.1

Version 1.3.8

- [NEW] Brake Assist optional X and Y axis
- [NEW] SSCManager - support for additive scenes
- [NEW] Sample Component for Arcade Variable Flight Acceleration
- [FIXED] Input Control 2.5D may not return to original plane when bumped off course
- [IMPROVED] ShipControlModule - debug shows x and y local velocities

For the full change log, see Version History at the end of the manual.

Support Policy

For free support we will investigate reproducible bugs in our code. We may ask you to provide a simple scene with clear instructions on how to repro the issue. We can provide an upload area for the project files.

If this issue is critical to an announced game release date, we give it high priority. We also help with fleshing out new features that can improve game-play and that we could add to a new version. In addition, we offer customer support for discovering existing features and how to configure them, both in our [Unity forum](#) or on our [Discord channels](#).

To add "polish" to a game or general help with implementing our products (and even writing custom game-play code) we negotiate a flexible hourly rate which can be time-boxed to fit the studio or indie budget.

For ad-hoc on-going support, and to help us to keep supporting Sci-Fi Ship Controller for a long time, please support us on <https://www.patreon.com/scsmmedia>

We may alter this support policy from time to time without notice.

Getting Started

SSC comes bundled with a number of demo scenes and prefabs to get you started quickly. The demo scenes are set up with a single player ship, with keyboard input, to allow you to simply open the scenes in the Unity editor and hit play.

SSC has a lot of flexibility in dealing with many, many different scenarios. However, if you have a specific use-case in mind, the demo scenes may help you select a starting Ship Control Module configuration. Most people will want to start with one of the ship prefabs from the `SCSM\SciFiShipController\Prefabs\Ships` folder. Each of these prefabs has a basic ship model attached so you can quickly test it in your scene. You can add your own space craft models later.

After dropping one of the SSC ship prefabs into the scene, optionally drop in the PlayerCamera prefab (`SCSM\SciFiShipController\Prefabs\Environment`) and/or Celestials (`SCSM\SciFiShipController\Demos\Prefabs`). Both of these require a small amount of configuration. You will see warnings in the Unity Console at runtime in the Unity Editor if you forget to configure them.

Throughout this manual we refer to anything that can be flown by a player or an AI-player as a “ship”. All “ships” need to include a Ship Control Module. It could be any of the following:

- Aircraft / airplane which flies within a planet’s atmosphere or is affected by gravity
- Hover ship / craft or land-based speed racer
- Spaceship that travels through empty space or around a planet
- Space plane (acts like an airplane while inside a planet's atmosphere, but acts like a spaceship while in space)
- Fast 1-2-seater fighter
- Large space battle cruiser which includes turret-like weapons

Videos and Tutorials

Name	URL
Trailer (for the 1.3.0 release)	https://youtu.be/ApzeVodi-YI
Tech Demo 2 Trailer 1	https://youtu.be/U2s6ttNDcr8
Getting Started / Setup Basics Tutorial	https://youtu.be/-lvOhk_4P6k
Control Tab Basics Tutorial	https://youtu.be/8mNqLeun5eQ
Damage Basics Tutorial	https://youtu.be/6tfwtm9b1oU
Weapon Basics Tutorial	https://youtu.be/llpttO-UUyE
Physics-Based Tutorial	https://youtu.be/EPbzfG_2Ltw
Radar Basics Tutorial	https://youtu.be/1FMixzYXxuo
Docking Basics Tutorial	https://youtu.be/poqPosakse0
VR Flight Setup with SSC + S3D	https://youtu.be/m1OdRGvDOS0
Jump Gate Tutorial	https://youtu.be/_IN0_gfRe78
Mars Re-entry (from Tech Demo 4)	https://youtu.be/l0KVyijDHIU
Anti-Gravity Racer Demo	https://youtu.be/reYNcztvt3I
SSC with Unity Mega City Tech Demo **	https://youtu.be/flb8Xhl_v5g
SSC Asteroids Demo	https://youtu.be/zYNTHua6odY
Ship AI System (3-part series)	https://forum.unity.com/threads/594448/page-2#post-5368236

** Unity Mega City assets are not included with Sci-Fi Ship Controller. This Tech Demo is only to show that SSC can be integrated with other systems.

Demos

General Demos

There are nine demo scenes available which can be found by navigating to the folder SCSM\SciFiShipController\Demos\Scenes. With the exception of the Asteroids and Docking demos, they contain one ship which can be controlled by the player (just run the scene to start the demo). Apart from the "Endless Flier Demo", each demo uses the same basic control scheme from the legacy Unity input system:

- Up arrow key is forwards thrust, down arrow key is the brakes
- A/D keys are used to turn left and right
- W/S keys are used to pitch up and down
- Left and right arrow keys are used to roll clockwise and anticlockwise
- Additionally, in the "Explorer (Physics) Demo", Q/E keys are used to move up and down

If you'd like to use a game controller with the demo scenes or the new Unity Input System, see the section called "Player Input Module" later in this manual.

The Asteroids Demo and Docking Demo scenes are entirely controlled by the Ship AI system.

In the City Demo scene, the camera can be switched between squadrons with the "Y" key. To switch ships in the same squadron, press the "T" key.

For **Demo Scripts**, look in the "Runtime and API" chapter later in this manual.

AG SSC Racer

The first tech demo, AG SSC Racer, was released on PC, Xbox, and Mac. It can be found on the Microsoft Store or can be downloaded from our website at <http://scsmmedia.com/ssc.html>. The game was built using the Anti-gravity Racer Demo scene with some additional game play elements added.

Tech Demo 2

The ninth demo, can be found in the Demos\TechDemo folder. In the Unity forums this is known as Tech Demo 2 as it is the second technical demo release. Unlike Tech Demo 1, the full game is included with Sci-Fi Ship Controller.

Before playing the scene there are a few things to do:

1. Set the Unity User Layer 27 to "Small Ships" (if you need to change the layer number because it is already in use, in the TechDemo scene, click on the "TechDemo2 Controller" and change the "Small Ships Unity Layer" number in the Unity Inspector.
2. In Project Settings, Player, under "Other Settings" set the Rendering Color Space to "Linear"

More information about Tech Demo 2 can be found in the following Unity forum posts:

<https://forum.unity.com/threads/594448/page-6#post-6277157>

<https://forum.unity.com/threads/594448/page-6#post-6324246>

This tech demo was also released as a game on PC and Xbox. You can read more about that here:

<https://forum.unity.com/threads/977508/>

To convert the Tech Demo over to using the (new) Unity Input System, see the readme in the scene folder.

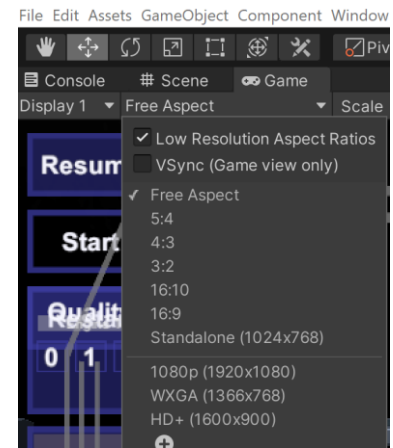
Tech Demo 3

New in SSC version 1.3.0, this demo brings several features together. When used with Sticky3D Controller (our character controller which is also available on the Asset Store), it provides an immerse experience packed with technical examples of how to solve common problems when building Sci-Fi space sims. The action takes place while docking a small space shuttle at a space port. To use with Sticky3D, download the package from the Beta Program or follow the instructions at the top of the TechDemo3.cs script.

Most of the action is controlled by the TechDemo3 script which is attached to the “TechDemo3Controller” gameobject in the scene.

The basic game play goes something like this:

1. If Sticky3D Controller is not installed, always use the Walk Thru option
2. If Sticky3D Controller is installed, use the Walk Thru option if it is enabled, otherwise use the Sticky3D character as the player (this gives the richest interactive experience but requires Sticky3D, which is available as a separate asset from the Unity Asset Store).
3. If in Walk Thru mode, wait for the user to press the “I” key to commence the docking procedure.
4. If using Sticky3D, sit the player down in the Shuttle cockpit seat and wait for the player to select the “docking” button on the console.
5. Along the way, display an appropriate action message on the Heads-Up Display. E.g., “Initiate docking procedures”, “Make you way to Space Port Services” etc.
6. The Walk Thru uses a camera “ship” which literally flies through the scene. The camera is an AI ship that follows the appropriate path. Each path was set up in our SSCManager path editor. It takes advantage of Door Proximity components etc to let it pass though doors. It even “knows” how to take the lift to the top deck of the space port.
7. With Sticky3D, the user can interact with door and lift controllers in the scene. The user can toggle between first and third person using the “V” key. Movement is the “standard” WSAD and mouse controls look. This is for convenience in the demo. In a real game you could configure it to use whatever suited your game setup. Also, you could use the “new” Unity input system or various other setups (even VR if you wanted too!).
8. After exiting the small shuttle and entering the space port, the player is encouraged to go into the Services room. If Sticky3D is in play, an assistant tries to get your attention. In Walk Thru (without Sticky3D installed), there is an empty seat behind the services desk.
9. Stuff now gets interesting and all manner of things start to happen. Getting to the top deck of the space port is indicated and the obvious method is via the lift.
10. If Sticky3D is in play, the character should call the lift and take the lift to the top deck. In Walk Thru mode, this happens automatically.
11. If Sticky3D is in play, you will need to put on your helmet to get out the outer doors (it’s difficult to breath in space).
12. As the player walks across the top deck to their awaiting Hawk fighter, the enemy fighters start attacking the space station itself. Thankfully, your friendly fighters have already started to engage the invading force.
13. The player should climb up the ladder and get into the cockpit of the Hawk (this happens automatically in Walk Thru mode).
14. When the player gets to the top of the ladder, we switch control from either the Sticky3D character or the camera “ship” to the Hawk fighter. We power up the ship and the rest is up to you.
15. Now the player (that’s you!) needs to destroy the enemy fighters and save the space port.



If you have a **low-powered device** (CPU and/or GPU), the demo will always work better in a build. However, if you want to run it in the editor, reducing the screen resolution of the Game view can make a big difference. Tick the “Low Resolution Aspect Ratios” can help.

Most games are render-bound. That is, drawing all the pixels on the screen, often multiple times for the same “final” image, can be “expensive”.

By reducing the number of pixels drawn, you can increase the in-editor performance.

Tech Demo 4

This technical demo is still in development. We continue to develop features for TD4 that will appear in SSC as they are ready for release. Although we’re hoping to release TD4 around version 1.5.0, the overall aim is to create n immersive interactive experience on Mars with full integration with our character controller (Sticky3D Controller). The focus in on the underlying tech, not the artwork or 3D models.

We want to help you solve game development issues for sci-fi related games.

Ship Control Module

Ship Control Module – Overview

This module enables you to implement ship behaviour on the object it is attached to. It contains the core configuration required to get your ship flying.

The module includes the following configuration categories. They appear in the logical order you’ll typically set up a ship.

Category	Purpose
Physics	Where you select (realistic) Physics mode or Arcade mode. Also determines how gravity and mass affects a ship’s behaviour.
Control	Includes options for over-riding the player’s input behaviour.
Thrusters	Used to control the ship by adding forces to move the ship's position and torques to rotate the ship. Can also add Particle System effects and audio effects.
Aero	Determine how your ship interacts with the air around it (its aerodynamic properties).
Combat	Includes how the ship takes damage, how the ship is re-spawned, how weapons are configured, and how a ship is identified.

In-Scene Editing

Sci-Fi Ship Controller has a number of features that support in-scene editing for more intuitive development. This enables you to select different attributes of your ship and move them around within the scene view. The values in the SSC module editors will automatically be updated.

In the Ship Control Module editor, whenever you see a small “G” button it indicates that the Gizmos in the scene for that feature can be shown or hidden. Some items like Centre of Lift and Direction, and Centre of Thrust and Thrust Direction are non-selectable. Others like Centre of Mass, Wings, Control Surfaces, Thrusters, and Weapons, are selectable and can be manipulated in the scene view.

To edit the location, size, and/or rotation of an item associated with a ship perform the following actions:

1. In the scene Hierarchy, select the parent gameobject of the ship
2. In the Inspector, navigate to the appropriate tab in the Ship Control Module and make sure the Gizmos are enabled. If in doubt, click the “G” button in the editor to toggle them on/off.
3. In the scene view, move your mouse over the gizmo for the item and click it to select or unselect it.
4. Move, Rotate, or Resize the component with the standard Unity Editor tools. Note: Not all items have all 3 manipulation methods enabled. For example, it doesn’t make sense to be able to rotate the ship’s centre of mass however it does for a control surface or wing.

Physics Tab

The Physics tab is usually a good place to start when setting up a ship, as it contains a lot of the basics. Here you can specify the mass of the ship, as well as the strength and direction of gravity acting upon the ship. You can also specify whether Unity should set the centre of mass of the ship automatically or if you want to set it manually.

Probably the most important value to start with is the one at the top: The physics model. The physics model determines what options are available for ship control and behaviour. There are two options currently available: Physics-based and Arcade.

Physics-based mode is best employed for games aiming to achieve a large degree of realism, or at the very least evoke a sense of realism from players. In general, only physically realistic options are available. Ships can only be turned using thrusters and control surfaces.

Arcade mode, on the other hand, provides a number of extra options to enhance ship feel and gameplay while removing certain behaviours entirely in order to make ship setup and control easier. For example, pitch/roll/yaw acceleration and turn acceleration options are only available in arcade mode. Pitch/roll/yaw acceleration values directly determine how quickly the ship rotates on each axis - in physics-based mode rotations can only be achieved indirectly via the use of thrusters. Turn acceleration adds force inputs to a ship based on its motion in order to make the ship's velocity more in line with the way it is facing, which is common in a lot of flight games as it makes flight control and movement a lot more intuitive for players.

Property	Description
Initialise On Awake	If enabled, the InitialiseShip() will be called as soon as Awake() runs for the ship. This should be disabled if you are instantiating the ship through code.
Physics Model	The physics model determines which options are available for ship control, as well as how physics for certain things such as thrusters is handled. Select Physics-Based mode to create physically realistic ships and aircraft, or select Arcade mode to replicate the behaviour of more fictitious craft.
Mass	The property of 'mass' determines how heavy the ship is (in kilograms). The point through which all of the mass of the ship seems to act is known as the 'centre of mass' (CoM) and is indicated by the grey sphere in the scene view. When 'Set CoM Manually' is disabled, the centre of mass will be automatically determined by Unity based on the position of any colliders attached to the rigidbody. Otherwise, the centre of mass can be adjusted by modifying the value of 'Centre of Mass' in the inspector or by selecting the grey sphere in the scene view and dragging it with the move tool. The 'Reset Centre Of Mass' button resets the centre of mass to the position automatically determined by Unity.
Set CoM Manually	When enabled, the centre of mass (the point through which all forces on the ship act) can be edited manually. When disabled, the centre of mass will be positioned at the default position specified by Unity - this is determined by the ship's colliders.
Centre of Mass	The position of the centre of mass in local space.

Property	Description
Gravity	Acceleration changes the strength of gravity. Increasing it increases the pull of gravity. Direction changes the direction in which gravity acts. This direction is indicated by the grey arrow in the scene view, and can be adjusted by selecting the grey sphere and using the rotation tool.
Acceleration	The acceleration due to gravity in metres per second squared. Earth gravity is approximately 9.81 m/s ² .
Direction	The direction in which gravity acts on the ship in world space.
Pitch/Roll/Yaw Acceleration	Use Pitch/Roll/Yaw Acceleration to determine how quickly the ship can turn on each axis.
Roll Acceleration	How fast the ship accelerates when rolling left and right in degrees per second squared. Increasing this value will increase how fast the ship can be rolled left and right by pilot and rotational flight assist inputs.
Yaw Acceleration	How fast the ship accelerates when turning left and right in degrees per second squared. Increasing this value will increase how fast the ship can be turned left and right by pilot and rotational flight assist inputs.
Turn Acceleration	Use turn acceleration to add a more 'arcade' feel to ship movement. Flight Turn Acceleration adds force inputs to cause the ship to move in the direction it is facing while in the air, while Ground Turn Acceleration does the same while the ship is near the ground.
Flight Turn Acceleration	How quickly the ship accelerates in metres per second squared while in the air to move in the direction the ship is facing.
Ground Turn Acceleration	How quickly the ship accelerates in metres per second squared while near the ground to move in the direction the ship is facing.

Control Tab

For some ships, the control setup is very simple, while in others (such as the classic hover ship) the control setup is more complicated. Essentially, the Control tab is used for setting up any input control that the computer will do instead of the player. This includes input assists (such as the rotational flight assist) which make flight easier for players, as well as control modifiers that allow more interesting behaviour to occur (such as stick to ground surface, which controls the pitch, roll and vertical inputs of the ship to allow it to orient itself to the ground surface and maintain a given distance from it). The Control tab determines a lot of the behaviour of the ship related to gameplay.

Property	Description
Rotational Flight Assist	Rotational Flight Assist helps a pilot to control a ship by applying axial inputs to oppose rotational velocity and slow down spinning motion when the pilot releases the input on that axis.
RFA Strength	Increasing this value will increase how quickly rotational flight assist slows down spinning motions. Setting it to zero will disable rotational flight assist entirely.
Translational Flight Assist	Translational Flight Assist helps a pilot to control a ship by applying translational inputs to oppose movement on non-forward axes when the pilot releases the input on those axes, which aligns the ship's velocity more with its forward direction, making turning easier and more intuitive.
TFA Strength	Increasing this value will increase how quickly translational flight assist slows down movement in a particular direction. Setting it to zero will disable translational flight assist entirely.

Property	Description
Stability Flight Assist	Stability Flight Assist helps a pilot to control a ship by applying rotational inputs to keep the ship stable at its current orientation when the pilot releases rotational inputs.
SFA Strength	Increasing this value will increase how rigidly stability flight assist keeps the ship stable. Setting it to zero will disable stability flight assist entirely.
Brake Flight Assist	Brake Flight Assist helps a pilot to slow a ship when the pilot releases the input on the forward or backward (z) axis. At slow speeds it can bring a ship to a complete stop. This is overridden in forwards direction when AutoCruise is enabled on PlayerInputModule.
BFA Strength Z	Strength of the brake flight assist on local z-axis. Set to zero to disable [Default = 0]. Operates on Forward and Backward movements when there is no ship input.
Min. Speed (m/s)	The effective minimum speed at which the brake flight assist will operate. [DEFAULT -10m/s]
Max. Speed (m/s)	The effective maximum speed at which the brake flight assist will operate. [DEFAULT +10m/s]
BFA Strength X	Strength of the brake flight assist on local x-axis. Set to zero to disable [Default = 0]. Operates on left and right movements when there is no ship input.
BFA Strength Y	Strength of the brake flight assist on local y-axis. Set to zero to disable [Default = 0]. Operates on up and down movements when there is no ship input.
Brake Axis X	Brake Flight Assist will also operate on the local x-axis (left and right).
Brake Axis Y	Brake Flight Assist will also operate on the local y-axis (up and down).
Limit Pitch/Roll	When Limit Pitch/Roll is enabled, the ship is limited to a range of pitches (as specified by the Max Pitch) and roll is controlled by yaw input, between a small range of roll angles (as specified by Max Turn Roll). This can either be constrained to the world upward direction (as would be used in an arcade airplane game) or constrained to the ground surface (as would be used in a hover-racing game) by enabling Stick To Ground Surface.
Max Pitch	The maximum pitch in degrees that the ship is allowed to attain.
Pitch Speed	How fast the ship pitches in degrees per second.
Max Turn Roll	The maximum roll in degrees that the ship is allowed to attain.
Turn Roll Speed	How fast the ship rolls in degrees per second.
Roll Control Mode	How roll is controlled. When Yaw Input is selected, roll is determined by the yaw input (turning left will make the ship roll left and vice versa). When Strafe Input is selected, roll is determined by the strafe input (strafing left will make the ship roll left and vice versa).
Responsiveness (Pitch/Roll)	How fast the ship pitches/rolls to match the target pitch/roll (for instance the ground surface if Stick To Ground Surface is enabled).
Avoid Ground Surface	Helps the ship to avoid crashing into the ground (CURRENTLY in Technical Preview). It overrides vertical input by applying force when below the perpendicular target ground distance using a PID controller.
Stick To Ground Surface	When Stick To Ground Surface is enabled, the ship will orient itself and maintain a relatively constant distance to the ground below it, as specified by the Target Distance.

Property	Description
Orient Upwards In Air	Enable this to revert to the default behaviour of Limit Pitch/Roll (constraining the ship to a limited range of pitch and roll, facing upwards) when there isn't a detectable ground surface below the ship (i.e., when the ship is high up in the air).
Ground Match Smoothing	Whether the movement used for matching the Target Distance from the ground is smoothed. Enable this to prevent the ship from accelerating too quickly when near the Target Distance.
Look Ahead	When Stick To Ground is enabled, whether the ground match algorithm 'looks ahead' to detect obstacles ahead of the ship.
Target Distance	When Stick To Ground is enabled, the distance the ship will attempt to maintain to the ground surface below it.
Target Distance Above	When Above Ground Surface is enabled, the distance the ship will attempt to maintain above the ground surface below it.
Min Distance	When Ground Match Smoothing is enabled, the minimum distance the ship will attempt to maintain to the ground surface below it.
Max Distance	When Ground Match Smoothing is enabled, the maximum distance the ship will attempt to maintain to the ground surface below it.
Max Check Distance	The max distance the ship will check below it to find the ground surface. When the ship is further than this distance away from the ground surface it will either revert to the default behaviour of Limit Pitch/Roll (if Orient Up In Air is enabled) or it will revert to standard six-degrees-of-freedom behaviour (if Orient Up In Air is disabled).
Responsiveness (Target Distance)	How responsive ship is to sudden changes in the distance to the ground. Increasing this value will allow the ship to match the Target Distance more closely but may lead to a juddering effect if increased too much.
Damping	How much the up/down motion of the ship is damped when attempting to match the Target Distance. Increasing this value will reduce overshoot but may make the movement too rigid if increased too much.
Max Acceleration (Target Distance)	The limit to how quickly the ship can accelerate to maintain the Target Distance to the ground. Increasing this value will allow the ship to match the Target Distance more closely but may look less natural.
Ground Normal Calc.	How the normal direction (orientation) is determined. When Single Normal is selected, the single normal of each face of the ground geometry is used. When Smoothed Normal is selected, the normals on each vertex of the face of the ground geometry are blended together to give a smoothed normal, which is used instead. Smoothed Normal is more computationally expensive.
Ground Layer Mask	Only geometry in the specified layers will be detected as being part of the ground surface.
Ground Normal History	The number of past frames (including this frame) used to average ground normals over time. Increase this value to make pitch and roll movement smoother. Decrease this value to make pitch and roll movement more responsive to changes in the ground surface.
Roll Power	How much power of the ship's roll thrusters is used to execute roll manoeuvres. Increasing this value will increase the roll speed and responsiveness of the ship.
Pitch Power	How much power of the ship's pitch thrusters is used to execute pitching manoeuvres. Increasing this value will increase the pitch speed and responsiveness of the ship.

Property	Description
Yaw Power	How much power of the ship's yaw thrusters is used to execute yaw manoeuvres. Increasing this value will increase the yaw speed and responsiveness of the ship.
Input Control (2.5D)	Limit an input axis to achieve 2.5D-like flight behaviour. This is helpful when you have a 3D environment but would like the ship to be constrained on one axis.
Input Control Limit	The target value of the X or Y axis
Input Moving Rigidness	The rate at which force is applied to limit control
Input Turning Rigidness	The rate at which the ship turns to limit or correct the rotation
Input Forward Angle	The forward X angle for the plane the ship will fly along. Only applies when the Input Control Axis is X.

Thrusters Tab

The Thrusters tab is exactly what it sounds like: It's where you set up your ship's thrusters. Each thruster has a vector direction and a force amount specified, and is linked to the various player inputs via the selection of which force direction and (in physics-based mode) moment directions (read: rotations) the thruster is able to move the ship in (this can be set up automatically using the auto-populate forces and moments button).

The Gizmos in the scene point in the direction that the thrust will be applied. For example, a forward thruster will point backward as thrust is applied backwards to propel the ship forwards.

An effects object can also be specified for each thruster to link it with effects to be triggered when the thruster is in use.

As more thrust is applied (either from a human player via the Input Player Module, or in code via an AI-player), the Particle System effect increases. If an Audio Source is attached to the effect, the volume increases or decreased based on the amount of thrust that is being applied.

Property	Description
Thruster Systems Started	Are the thruster systems online or in the process of coming online?
Startup Duration	The time, in seconds, it takes for the thrusters to fully come online.
Shutdown Duration	The time, in seconds, it takes for the thrusters to fully shutdown.
Central Fuel	For thrusters, use a central fuel level, rather than fuel level per thruster.
Central Fuel Level	The amount of fuel available to the whole ship when "Central Fuel" is enabled - range 0.0 (empty) to 100.0 (full).
FX when Stationary	When the ship is enabled, but ship movement is disabled, should thrusters fire if they have Min FX Always On enabled?
Name	The name of the thruster
Max Thrust (kN)	The max thrust in kilonewtons this thruster can generate.
Throttle	The amount of available power being supplied to the thruster.
Relative Position	The position of the thruster in local space.
Thrust Direction	The direction of thrust of the thruster in local space.
Thrust Input	Which input activates this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).

Property	Description
Primary Moment Input	The rotational input which is primarily used to activate this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).
Secondary Moment Input	The rotational input which is secondarily used to activate this thruster. This determines what input/s are linked with this thruster (if you don't know how to use this, the Auto-Populate Forces and Moments button can be used to calculate this automatically).
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this thruster (i.e., what performance level the thruster will have when its health reaches zero). The performance level affects how much thrust this thruster generates.
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.
Effects Object	The object which has the effects that should be enabled when the thruster is used (e.g., sounds, particle effects, etc.) attached. Set the Volume of the AudioSource to represent the maximum volume at full thrust. This should typically be a child gameobject of the ship's parent gameobject.
Minimum Effects Rate	The 0.0 to 1.0 value that indicates the minimum normalised amount of any particle effects or audio sources that are applied when a non-zero thrust input is received for this thruster. The default is 0 which will apply a linear particle emission rate or audio volume based on the amount of thrust input received. If the full particle emission rate or audio volume should be applied when any input is received, set the value to 1.0.
Min FX Always On	When Minimum Effects Rate > 0 and Throttle > 0 the effects fire when thruster input is 0. The Limit FX On Y and Z settings are still honoured when this is true.
Limit FX on Y Axis	Limit when the effects are used for this thruster based on the speed of the ship along the local Y axis (up or down)
Min. FX on Y Axis	The minimum speed in m/s on the local y-axis the ship must be travelling at before the effects will activate.
Max. FX on Y Axis	The maximum speed in m/s on the local y-axis the ship can be travelling for the effects to be active.
Limit FX on Z Axis	Limit when the effects are used for this thruster based on the speed of the ship along the local Z axis (forward or backward)
Min. FX on Z Axis	The minimum speed in m/s on the local z-axis the ship must be travelling at before the effects will activate.
Max. FX on Z Axis	The maximum speed in m/s on the local z-axis the ship can be travelling for the effects to be active.
Throttle Up Time	The length of time, in seconds, it takes the thruster to go from no thrust to maximum thrust. [Default = 0 or instant thrust increase]. NOTE: Full thrust will be reached before the throttle up time as the curve is flattened towards the min and max values.
Throttle Down Time	The length of time, in seconds, it takes the thruster to go from maximum thrust to no thrust. [Default = 0 or instant thrust decrease]. NOTE: Zero thrust will be reached

Property	Description
	before the throttle down time as the curve is flattened towards the min and max values.
Fuel Level	The amount of fuel available - range 0.0 (empty) to 100.0 (full). Only applies when Central Fuel is disabled.
Fuel Burn Rate	The rate fuel is consumed per second. If rate is 0 (the default), fuel is unlimited.
Heat Level	The heat of the thruster or engine - range 0.0 (starting temp) to 100.0 (max temp). At 100, the thruster will produce no thrust.
Heat Up Rate	The rate heat is added per second. If rate is 0, the heat level never changes.
Cool Down Rate	The rate heat is removed per second. This is the rate the thruster cools when not in use. Has no effect if Heat Up Rate is 0.
Overheat Threshold	The heat level that the thruster will begin to overheat and start producing less thrust.
Burnout on Max Heat	When the thruster reaches max heat level of 100, will the thruster be inoperable until it is repaired?

Aero Tab

The Aero tab is where you determine how your ship interacts with the air around it (its aerodynamic properties). This can be through altering environmental properties (the medium density), the drag properties (how moving through the air slows the ship down on each axis) or by adding wings and control surfaces. Wings simulate the effect of parts moving through the air generating lift (upwards force); their properties are controlled by changing the size and angle of attack (inclination) of the wing, as well as the stall effect (how much the effect of stalling affects the wings of the ship). Control surfaces are moving parts that change the aerodynamic properties of the ship in order to let the pilot control it, such as ailerons, rudders and air brakes. Control surfaces are automatically linked to the correct player inputs based on the type of control surface and where it is positioned relative to the centre of mass of the ship.

The drag properties of the ship determine how the airflow around the ship affects the movement of the ship. After making any mesh changes, click the Calculate Drag Properties to recalculate the internally stored drag properties of the ship. Then use the Drag X/Y/Z Coefficients to alter how much drag the ship has on each axis. More streamlined axes of the ship should have a lower drag coefficient while flatter axes should have a higher drag coefficient. In Arcade mode, you can also use the Angular Drag Factor to alter how quickly angular drag will slow down any spinning motion and Disable Drag Moments to prevent the ship from rotating due to moments caused by drag.

NOTE: If you had some colliders disabled, after clicking "Calculate Drag Properties", you will need to disable them again.

Property	Description
Medium Density	The medium density property defines the density of the medium (in kilograms per cubic metre) the ship is travelling through (generally in air). In less dense atmospheres this value should be lower and in more dense atmospheres this value should be higher. At low altitudes in Earth's atmosphere the value is approximately 1.293 while in space (where there is virtually no air) it should be set to zero to achieve realism. The medium density affects all aerodynamics (i.e., drag, lift, etc.).
Drag X Coefficient	The coefficient of drag of the ship on the x-axis. Increasing the coefficient of the drag will increase the effect of drag.

Property	Description
Drag Y Coefficient	The coefficient of drag of the ship on the y-axis. Increasing the coefficient of the drag will increase the effect of drag.
Drag Z Coefficient	The coefficient of drag of the ship on the z-axis. Increasing the coefficient of the drag will increase the effect of drag and reduce the top speed in the forwards and back directions.
Angular Drag Factor	How strong the effect of angular drag is on the ship. Setting this to 1 will make it physically realistic.
Disable Drag Moments	This will prevent drag causing the ship to rotate.
Drag X (Moment) Multiplier	A multiplier for drag moments causing rotation along the local (pitch) x-axis. Decreasing this will make these moments weaker.
Drag Y (Moment) Multiplier	A multiplier for drag moments causing rotation along the local (yaw) y-axis. Decreasing this will make these moments weaker.
Drag Z (Moment) Multiplier	A multiplier for drag moments causing rotation along the local (roll) z-axis. Decreasing this will make these moments weaker.

Wings

Property	Description
Wing Stall Effect	How much the effect of stalling affects ship flight. Setting this to zero will make the effect of stalling very minimal.
Name	Name of the wing. E.g. Front Wing, Tail Wing, Left Front Wing
Angle Of Attack	The angle of attack (inclination above the x-z plane) of the wing in degrees.
Length	The span (length on the z-axis) of the wing in metres.
Width	The chord (width on the x-axis) of the wing in metres.
Relative Position	The position of the wing in local space.
Lift Direction	The direction of the lift force of the wing in local space.
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this wing (i.e. what performance level the wing will have when its health reaches zero). The performance level affects how much lift the wing generates.
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.

Control Surfaces

Control surfaces allow you to simulate the effect of moving parts changing the lift and drag properties of the ship in order to control the movement of the ship. Control surfaces require the ship to be moving relative to air around it in order to operate.

Property	Description
Type	The type of control surface this is. The type determines how the control surface moves and what inputs control it. Ailerons control the roll of the ship, and are

Property	Description
	generally required to be placed symmetrically on opposite sides of the ship. Elevators control the pitch of the ship, and should generally be placed behind the centre of mass of the ship. Rudders control the yaw of the ship and should generally be placed in the middle of the ship (not to the left or to the right). Air brakes are used to slow the ship down.
Length	The span of the control surface in metres.
Width	The chord of the control surface in metres.
Relative Position	The position of the control surface in local space.
Damage Region	Which damage region this part belongs to.
Min Performance	The minimum possible performance level of this control surface (i.e. what performance level the control surface will have when its health reaches zero). The performance level affects how effective the control surface is.
Starting Health	The initial health value of this part. This is the amount of damage that needs to be done to the damage region this part is associated with for the part to reach its min performance.
Use Brake Component	Whether a brake component is used.
Brake Strength	The strength of the braking force.
Ignore Medium Density	Whether the strength of the brake force ignores the density of the medium the ship is in (assuming it to be a constant value of one kilogram per cubic metre).
Min Acceleration	The minimum braking acceleration (in metres per second) caused by the brake when the brake is fully engaged. Increase this value to make the ship come to a stop more quickly at low speeds.

Combat Tab

Includes how the ship takes damage, how the ship is re-spawned, how weapons are configured, and how it is identified to radar and others.

Damage

There are three damage modes: simple, progressive, and localised.

Damage Model	Description	Damage Regions
Simple	Ship has a single health value. An effects object, like an explosion prefab, can be assigned for when the health of the ship reaches 0. A Destruct object can be used to explode the ship into pre-made fragments.	Only 1 with or without shielding
Progressive	Allows the performance of certain ship parts (i.e., thrusters, wings, weapons) to decrease as the ship takes damage. Individual parts can optionally take Progressive Damage. They can have different Starting Health values. A Destruct object can be used to explode the ship into pre-made fragments.	Only 1 with or without shielding
Localised	Allows 1 or more parts to be assigned to a Damage Region of the ship. The performance of those parts in that region are decreased as the region takes damage.	1 or more regions with or without shielding.

Damage Model	Description	Damage Regions
	Each region can be assigned an effects object for when its health reaches 0.	

Progressive or localised damage can be assigned to parts including thrusters, wings, control surfaces and weapons.

Property	Description (Main Region)
Damage Model	Determines how damage is calculated and applied to the ship. In Simple mode, the ship has a single health value. The only effects of damage are "visual until the health value reaches zero, at which point the ship is destroyed and optionally respawns. In Progressive mode, as the ship takes damage the performance of parts is affected. In localised mode different parts can be damaged independently of each other.
Starting Health	How much 'health' the ship has initially.
Is Invincible	When invincible, it will not take damage however its health can still be manually decreased. When this main region is invincible, so is the whole ship.
Use Shielding	Whether the main damage region uses shielding. Up until a point, shielding protects the ship from damage (which can affect the performance of parts on the ship).
Recharge Rate	The rate per second that a shield will recharge. When the value is 0 (the default), the shield will never recharge.
Recharge Delay	The delay, in seconds, between when damage occurs to a shield and it begins to recharge.
Damage Threshold	Damage below this value will not affect the shield or the ship's health while the shield is still active (i.e., until the shield has absorbed 'amount' damage from damage events above the damage threshold).
Amount	How much damage the shield can absorb before it ceases to protect the ship from damage.
Col. Damage Resistance	Value indicating the resistance of the ship to damage caused by collisions. Increasing this value will decrease the amount of damage caused to the ship by collisions.
Controller Rumble	Whether controller rumble is applied to the ship by the ship control module.
Min Rumble Damage	The minimum amount of damage that will cause controller rumble.
Max Rumble Damage	The amount of damage corresponding to maximum controller rumble.
Damage Rumble Time	The time (in seconds) that a controller rumble event lasts for.
Effects Object	The particle and/or sound effect prefab that will be instantiated when the ship is destroyed. See also Effects Module.
Destruct Object	The destruct prefab that breaks into fragments when the ship is destroyed. When the ship is set to respawn, care should be taken around the settings of the Destruct Module and the Respawn timings. Consider what would happen if the user can still see the damaged (destruct) ship when the player ship has just respawned.
Use Damage Multipliers	Whether damage type multipliers are used when calculating damage from projectiles or beams. When projectiles with different "Damage Types" hit the ship, you can apply a relative amount of damage (based on the damage a projectile can normally inflict). For example, if a projectile can normally do 20 health points of damage, if that projectile had a damage type of say "Damage Type B" and your ship is pretty resilient to Type B projectiles, you could set the Damage Type B multiple for

Property	Description (Main Region)
	the ship to say 0.5 (50%). So, when the projectile hits the ship, it only does 10 health points of damage (20 points * 0.5 = 10 points of damage). See also "Damage Type" under Projectile Module or Beam Module.
Damage Type A-F	When Use Damage Multipliers is enabled, this is the relative amount of damage a projectile or beam with the matching Damage Type will inflict upon the ship.

Local Damage Regions can define areas of a ship that you want to receive individual damage. Examples could be "Engines" or "Bridge" or "Forward Weapons". It could also be a strategic area of your ship that is critical for completing missions like "(Food) Gallery", "Armory", "Hangar", "Landing Gear", "Tractor Beam" etc.

Property	Description (Localised Regions)
Name	The name of the damage region
Is Invincible	When invincible, it will not take damage however its health can still be manually decreased. If the main region is invincible, no localised region will receive damage either.
Relative Position	Position of this damage region in local space relative to the pivot point of the ship. Together with the size it determines what area the damage region encapsulates. This is the area that damage must occur at to impact this damage region.
Size	Size of this damage region (in metres cubed) in local space. Together with the relative position it determines what area the damage region encapsulates. This is the area that damage must occur at to impact this damage region.
Starting Health	How much 'health' the damage region has initially.
Use Shielding	Whether this damage region uses shielding. Up until a point, shielding protects the ship from damage (which can affect the performance of parts on the ship).
Damage Threshold	Damage below this value will not affect the shield or the damage region's health while the shield is still active (i.e. until the shield has absorbed 'amount' damage from damage events above the damage threshold).
Amount	How much damage the shield can absorb before it ceases to protect the damage region from damage.
Col. Damage Resistance	Value indicating the resistance of the damage region to damage caused by collisions. Increasing this value will decrease the amount of damage caused to the damage region by collisions.
Effects Object	The particle and/or sound effect prefab that will be instantiated when the damage region is destroyed. See also Effects Module.
Effect Follows Ship	The particle and/or sound effect will follow the damaged ship as it moves.
Destruct Object	The destruct prefab that breaks into fragments when the damage region's health reaches 0.
Child Transform	The child transform of the ship that contains the mesh(es) for this local region. If set, it is disabled when the region's health reaches 0. Setting this can also assist other weapons in the scene with determining Line-of-Sight to a damage region.
Damage Type A-F	When Use Damage Multipliers and Local Multipliers are enabled, this works the same as the Main Damage Region Damage Type multipliers – except it applies to each individual damage region.

Respawning

When a ship's health is reduced to 0 and it is destroyed, respawning options allow you to determine what happens next.

Property	Description
Respawning Mode	How respawning happens after the ship is destroyed. Options include "Don't respawn", "Respawn at Original Position", "Respawn at Last Position", "Respawn at Specific Position", "Respawn on Path"
Respawn Time	How long the respawn process takes (in seconds). This lets you delay the respawning so that the ship doesn't immediately reappear after being destroyed. You might want to allow enough time for an effect to run (like an explosion).
Respawn Position	The position in world space that the ship respawns from.
Respawn Rotation	The rotation in world space that the ship respawns with.
Col. Respawn Delay	The time (in seconds) between updates of the collision respawn position. Hence when the ship is destroyed by colliding with something, the ship respawn position will be where the ship was between this time ago and twice this time ago. Only relevant when Respawning Mode is set to "Respawn At Last Position".
Respawn Path	The Path from SSCManager that the ship will respawn when "Respawning Mode" is "Respawn On Path".
Respawn Velocity	The velocity in local space that the ship respawns with.
Stuck Time	The amount of time that needs to elapse before a stationary ship is considered stuck. When the value is 0, a stationary ship is never considered stuck. By default, this is set to 0 (Never). It may be useful for AI ships that get trapped or snagged on an obstacle. A ship will never become stuck when it has been disabled with DisableShip(..) or while respawning. To temporarily turn this off at runtime, set the StuckAction to "Do Nothing" in your code.
Stuck Speed Threshold	The maximum speed in m/sec the ship can be moving before it can be considered stuck. When a ship gets stuck it typically still trying to move and therefore has a non-zero speed. The default threshold is 0.1 m/sec.
Stuck Action	The action to take when the ship is deemed stationary or stuck.
<i>Do Nothing</i>	As the name suggests.
<i>Invoke Callback</i>	Create a custom method in your code that takes ShipControlModule as a parameter. Then set the shipControlModule.callbackOnStuck = YourCustomMethodName.
<i>Respawn On Path</i>	The ship will be respawned on the Path, closest to where it became stuck.
<i>Same As Respawning Mode</i>	Perform the action as indicated by the Respawning Mode.
Stuck Respawn Path	The Path from SSCManager that the ship will respawn on when Stuck Action is "Respawn On Path".

Weapons

Currently we support four kinds of ship-mounted weapons: Fixed Projectile canons and Turrets that fire Projectiles., and Beam (Fixed or Turret) weapons that fire ray or laser-style rays. Each weapon type can have multiple cannons/barrels/fire positions. All fire positions on the same weapon must fire in the same direction.

The first step to setup a Projectile weapon is to create a Projectile prefab. There are a few samples in `SCSM\SciFiShipController\Prefabs\Projectiles` to get you started. To create your own, use one of the samples as a template. Every Projectile requires a Projectile Module script attached to the parent gameobject. See the section called “Projectile Module” later in this manual.

For Beam weapons, you will need to create a Beam prefab. Samples can be found in the `SCSM\SciFiShipController\Prefabs\Beam` folder. To create your own, use one of the samples as a template or see the “Beam Module” section later in the manual.

Property	Description
Use Weapons when Movement is Disabled	When the ship is enabled, but movement is disabled, weapons and damage are updated. This can be helpful for say a capital ship that still wants to use turrets when it is not moving.
Name	Name of the weapon. E.g., Front Cannon, Front Right Turret
Type	The type or style of weapon. Current we support Fixed Projectile, Fixed Beam, Turret Projectile, or Turret Beam weapons. Turret Beam weapons are in Technical Preview.
Relative Position	The position of the weapon in local space relative to the pivot point of the ship. To visually modify this in the scene view, ensure the (G)izmo is turned on for this weapon, click the (F)ind button and move with the standard Unity Move Tool.
Multiple Fire Positions	If this weapon has multiple cannons or barrels
Fire Position Offsets	The positions of the cannon or barrel relative to the position of the weapon.
Fire Direction	The direction in which the weapon fires projectiles in local ship space. +ve Z is fire forwards, -ve Z is fire backwards. To visually modify this in the scene view, ensure the (G)izmo is turned on for this weapon, click the (F)ind button and rotate with the standard Unity Rotate Tool. If it is a turret weapon and the turret prefab or model is not facing forwards on the ship, the Fire Direction will need to be adjusted accordingly. The default is forwards (0,0,1).
Beam Prefab	Prefab template of the beam fired by this weapon. Beam prefabs need to have a Beam Module script attached to them.
Projectile Prefab	Prefab template of the projectiles fired by this weapon. Projectile prefabs need to have a Projectile Module script attached to them.
Reload Time	The minimum time (in seconds) between consecutive firings of projectile weapons.
Power-up Time	The minimum time (in seconds) between consecutive firings of beam weapons.
Max Range	The maximum distance (in metres) the beam weapon can fire.
Recharge Time	The time (in seconds) it takes the fully discharged beam weapon to reach maximum charge
Firing Button	The firing button or mechanism to use for this weapon. For a Player ship, this matches the Player Input Module. Fixed weapons can use the Primary or Secondary fire buttons from the Player Input Module, while Turrets also have the option of Auto-Fire.
Check Line of Sight	Auto-Fire turrets only. Whether the weapon checks line of sight before firing (in order to prevent friendly fire) each frame. Since this uses raycasts it can lead to reduced performance and should be used sparingly.
Unlimited Ammo	Can this (projectile) weapon keep firing and never run out of ammunition?
Ammunition	The quantity of projectiles or ammunition available for this weapon if there isn't unlimited ammo.

Property	Description
Charge Amount	The amount of charge the beam weapon has (0 = no charge, 1 = full charge)
Auto Targeting	When the Auto Targeting Module is attached, use this to indicate targets should be assigned to the weapon.
Damage Region	Which damage region this part belongs to.
Turret Pivot Y	For turret weapons, the transform of the pivot point around which the turret turns or rotates on the local y-axis
Turret Pivot X	For turret weapons, the transform on which the barrel(s) or cannon(s) elevate up or down on the local x-axis
Turret Min. Y	The minimum angle on the local y-axis the turret can rotate to
Turret Max. Y	The maximum angle on the local y-axis the turret can rotate to
Turret Min. X	The minimum angle on the local x-axis the turret can elevate to
Turret Max. X	The maximum angle on the local x-axis the turret can elevate to
Turret Move Speed	The rate at which the turret can rotate
Turret Inaccuracy	When inaccuracy is greater than 0, the turret may not aim at the optimum target position. This can improve the chances of the target not being hit by the weapon. It will have little or no effect if the weapon uses a guided projectile.
Turret Park Interval	When greater than 0, the number of seconds a turret will wait, after losing a target, to begin returning to the original orientation.
Heat Level	The heat of the weapon - range 0.0 (starting temp) to 100.0 (max temp).
Heat Up Rate	The rate heat is added per second for beam weapons. For projectile weapons, it is inversely proportional to the firing interval (reload time). If rate is 0, heat level never changes.
Cool Down Rate	The rate heat is removed per second. This is the rate the weapon cools when not in use.
Overheat Threshold	The heat level that the weapon will begin to overheat and start being less efficient. Beam weapons lose charge up to twice as fast when they are overheating.
Burnout on Max Heat	When the weapon reaches max heat level of 100, will the weapon be inoperable until it is repaired?

To create laser beam style weapons, use a Projectile Module prefab like ProjectileBasic5 which is included with SSC. It uses a Particle System component with a Trails item. To vary the length of the laser “beams”, modify the Particle System’s Start Speed.

Identification

Configurable in the editor or via code, identification can group ships into combat groups to help you identify them during gameplay.

Property	Description
Faction Id	The faction or alliance the ship belongs to. This can be used to identify if a ship is friend or foe. Neutral = 0.
Squadron Id	The (unique) squadron this ship is a member of. Do not place friendly and enemy ships in the same squadron.

Property	Description
Visible to Radar	<p>Is this ship visible to the radar system? If any ship (or Location) has this set, and the Radar system hasn't been added to scene, the SSC Radar system will be added automatically at runtime.</p> <p>Once the ship and the radar has been initialised, the ship will automatically begin sending packets of data to the radar system.</p>
Radar Blip Size	The relative size of the blip for this Location on the Radar mini-map

Player Input Module

This module enables you to map inputs from input sources to the six axes of the ship:

- Horizontal Input (left/right movement)
- Vertical Input (up/down movement)
- Longitudinal Input (forwards/backwards movement)
- Pitch Input (rotation on the local x-axis)
- Yaw Input (rotation on the local y-axis)
- Roll Input (rotation on the local z-axis)

It also enables you to map:

- two fire “button” inputs for the weapon system
- a “button” input for ship docking
- custom buttons to call your own code in Custom Player Input

The Player Input Module is very flexible and can take input from multiple sources including:

- Direct Keyboard (and mouse)
- Legacy Unity (input system)
- Unity Input System
- Unity XR
- Oculus API
- Rewired
- Vive

The Player Input Module script should be added to the parent gameobject of your player “Ship” prefabs along with a Ship Control Module script. If you are creating your own ship setup, we recommend copying the Player Input Module component from one of the ship prefabs included and “Paste Component as New” onto your ship parent gameobject. Modifying an existing Player Input Module component can be faster than starting from scratch.

At runtime in the Unity Editor, the Player Input Module has a Debug Mode which will help you identify what input data is being set to the Ship Control Module based on your configuration.

Each of the six axes, allow you to independently modify the sensitivity or responsiveness of the input coming from the various source systems. We recommend only enabling sensitivity on the required axes.

Property	Description
Initialise on Awake	If enabled, Initialise () will be called as soon as Awake () runs. This should be disabled if you want to control when the Player Input Module is enabled through code.
Enable on Initialise	Is input enabled when the module is first initialised? See also EnableInput() and DisableInput(). [Default: ON]
Custom Inputs Only	Are only the custom player inputs enabled when the module is first initialised? This could be useful when you want to get some menu or user options from the player without the ship moving when the scene first loads. See also DisableInput(allowCustomPlayerInput). [Default: OFF]

Direct Keyboard

When "Direct Keyboard" mode is selected, input can be modified by simply changing the key on the keyboard bound to each input.

Legacy Unity

When "Legacy Unity" is selected the Unity input manager will be used to receive input, requiring input axes to be set up in the Unity input manager and then referenced in the Player Input Module to bind input. For information on how to use the Unity input manager, see the Unity manual pages:

<https://docs.unity3d.com/Manual/ConventionalGameInput.html>

<https://docs.unity3d.com/Manual/class-InputManager.html>

Rewired

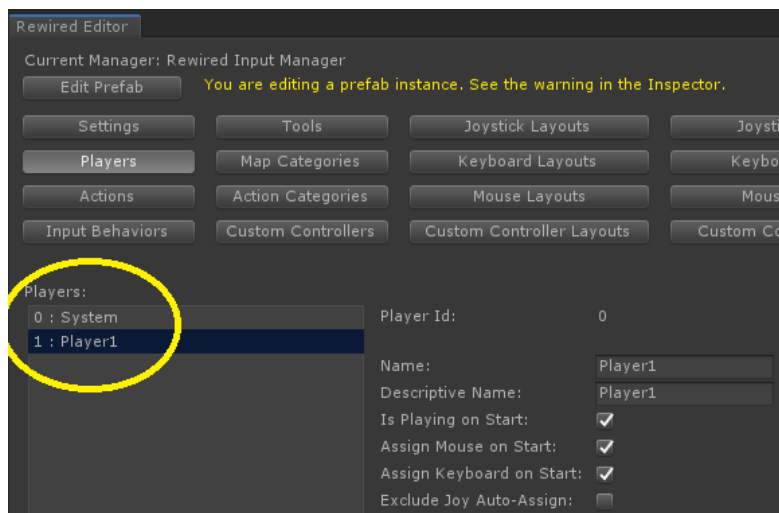
This popular 3rd party package supports a vast array of input controllers. To use Rewired with SSC you need to have separately purchased it from the Unity Asset Store and imported it into your project.

For more information on Rewired see:

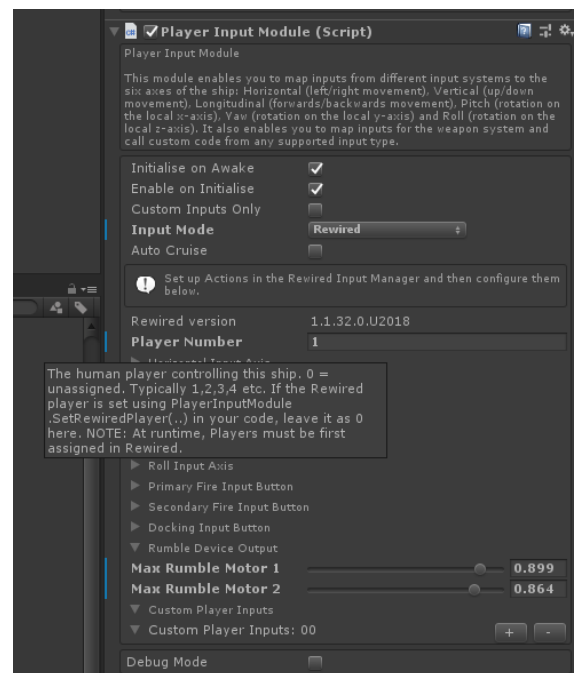
<https://assetstore.unity.com/packages/tools/utilities/rewired-21676>

Before configuring the Player Input Module with Rewired, you first need to setup the Rewired Input Manager in the scene. If you are not familiar with Rewired, here are the basic steps.

1. Add Rewired Input Manager to scene
2. Open Input Manager
3. Add a Player (and note the zero-based number in the list – e.g., 1, 2, 3 etc – System is typically 0)
4. Add Action Categories
5. Add Actions to those Categories
6. Add Joystick Maps (for gamepad use [T] Gamepad Template - see below)
7. Add Keyboard Maps
8. Add Joystick Maps to a Player (ensure one is set to Start Enabled)
9. Add Keyboard Maps to a Player (ensure one is set to Start Enabled)
10. For the Player, ensure Assign Keyboard on Start is enabled
11. In the SSC Player Input Module, set the Player Number to the player "number" from step 3 (0 mean unassigned).



IMPORTANT: Don't forget to create a Player and assign the "Player Number" in the Player Input Module.



For convenience, if you are using Unity 2019.1 with Rewired 1.25.2 or newer, we have included a quick setup prefab in the SCSM\SciFiShipController\Demos\3rdParty folder. Make sure you check out the Rewired_Readme file in the same folder first.

For PC with some kind of gamepad (e.g., Xbox One, Sony Dual Shock 4), we'd suggest setting up a Gamepad Template like in the table below.

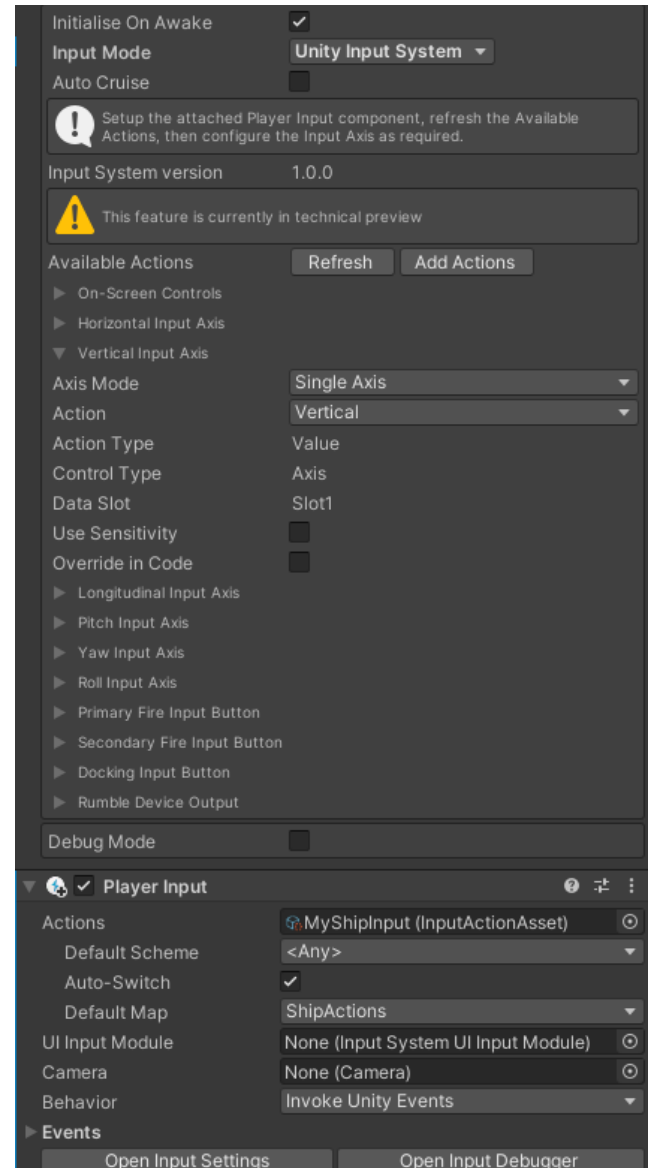
SSC Action	[T] Gamepad Template	Element Properties	Xbox One Controller	Xbox 360	Sony Dual Shock 4
Yaw	Left Stick X	Full	Left Stick X	Left Stick X	Left Stick X
Pitch	Left Stick Y	Full	Left Stick Y	Left Stick Y	Left Stick Y
Roll	Right Stick X	Full	Right Stick X	Right Stick X	Right Stick X
Vertical	Right Stick Y	Full	Right Stick Y	Right Stick Y	Right Stick Y
Longitudinal +ve	Right Shoulder 2	Range +ve, Contribution +ve	Right Trigger	Right Trigger	Right Trigger
Longitudinal -ve	Left Shoulder 2	Range +ve, Contribution -ve	Left Trigger	Left Trigger	Left Trigger
Primary Fire	Action Bottom Row 1		A Button		X Button
Secondary Fire	Action Top Row 2		Y Button		Triangle Button

The Player Input Module works for both Axis and Button input. It has been tested with Keyboard Maps and Joystick Maps in Rewired. If you see any issues, please let us know.

Unity Input System

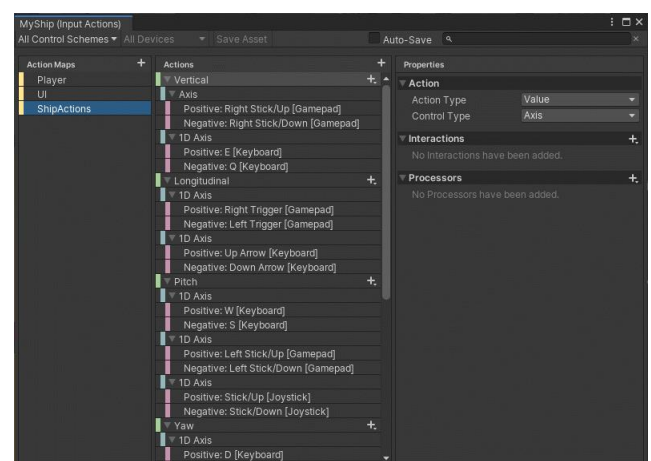
Sometimes referred to as the “New” Input System, v1.0.0 (preview) was first available in Unity 2019.1. Installed via the Unity Package Manager, this is an easy to setup, flexible system that supports many devices out of the box. To get started perform the following tasks:

1. Install Input System v1.0.0 or newer with the Unity Package Manager
2. Restart the Unity Editor
3. Add the Player Input component to the player ship
4. On the Player Input component, click “Create Actions...”
5. Give it a name and save the asset when prompted (e.g., MyShipInput)
6. The asset editor should be displayed. If not, select the new actions asset (e.g., MyShipInput) and click “Edit asset”.
7. Add a new Action Map (e.g., ShipActions)
8. Save the asset from the asset editor (and/or close it and save if prompted)
9. On the Player Input component attached to your ship prefab, set the Default Map to the one you just added (e.g., ShipActions) If the Actions says “None (Input Action Asset)” drag the asset (e.g., MyShipInput) into the slot first. Then set the Default Map. If the “Default Maps” doesn’t appear (see image on right), try running the scene or restarting Unity – this seems to be an issue with some versions of Unity Input System).
10. On the Player Input Module (attached to the same ship), change the Input Mode to “Unity Input System”
11. Resolve any configuration issues detected by the Player Input Module (this could include changing the default “Behaviour” on the Player Input component).
12. On the Player Input Module click “Add Actions” Now configure the various Input Axis in the Player Input Module like you would with any other Input Mode.



Actions in the Player input Module, map to Actions stored in a Unity Input Action asset (e.g., MyShipInput) that are stored in the Project. The Player Input component, not to be confused with Sci-Fi Ship Controller’s Player Input Module, contains a link to the Unity Input Action asset. Double-clicking on this will open the Unity Input Action editor. This is part of the (new) Unity Input System and is not part of SSC.

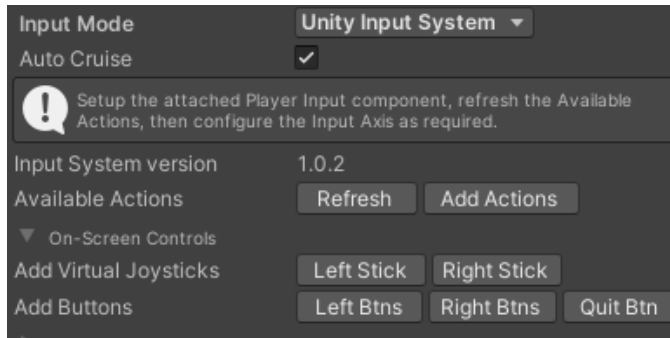
The Default Map setting (e.g., ShipActions) in the Unity Player Input component will determine which Actions are visible to the SSC Player Input Module. The Unity Input System can have multiple sets of Actions in what Unity



calls “Maps”. The SSC Player Input System will only use one of those maps.

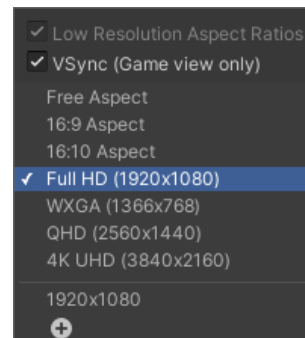
For more information on the Unity Input System see:

<https://github.com/Unity-Technologies/InputSystem/blob/develop/Packages/com.unity.inputsystem/Documentation~/index.md>



If you wish to use on screen controls for touch-enabled devices, we include some sample “**On-Screen Controls**” to get you started. You are free to create your own UI and touch input (see “Overriding Player Input Module in Code” below).

Before adding the On-Screen Controls, we’d suggest setting your Game view window to the approximate screen ratio that you’re going to (mostly) use.



When the first On-Screen Control is added to the scene, if an EventSystem is not already in the scene, one will be automatically added and highlighted. By default, the legacy “Standalone Input Module” script may be attached. If so, click “Replace with InputSystemUIInputModule”.

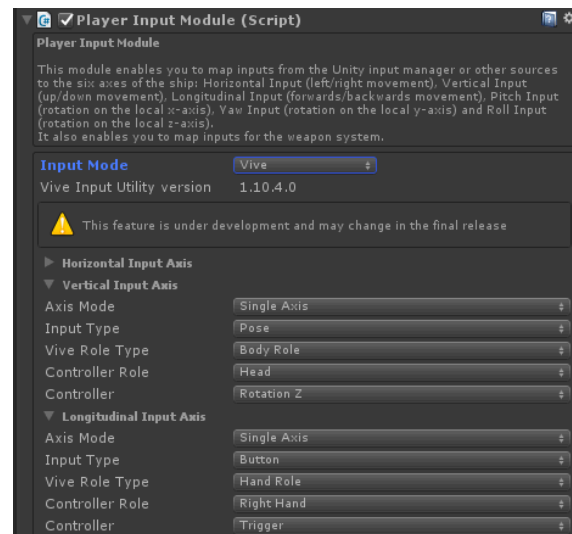
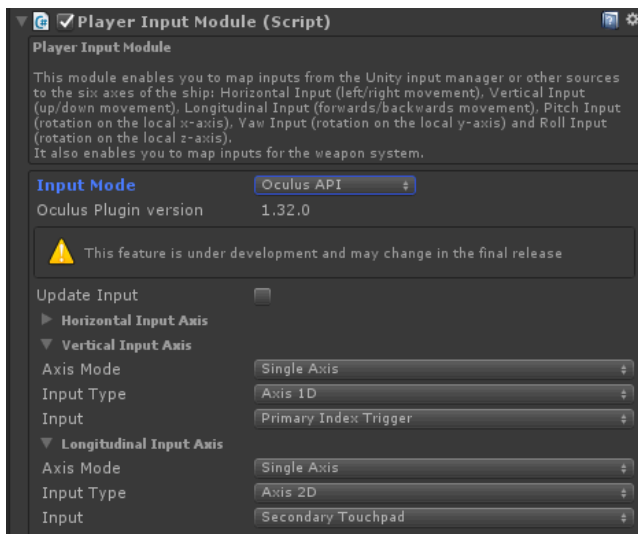
Virtual Reality (VR) Input

The Player Input Module can be configured to take input from a wide variety of VR controllers and devices. There are three options: Oculus API, Vive Input Module or the newer (recommended) Unity XR. The first two require some 3rd party components which are not included with SSC.

VR System	Description
Oculus API	<p>This requires the downloading and setup of the 3rd party Oculus package. https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022 See also https://www.oculus.com/setup/ https://docs.unity3d.com/Manual/OculusControllers.html SSC was tested with version 1.32.0.</p>
Vive Input Module	<p>This requires the downloading and setup of the 3rd party Vive package. https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219 SSC was tested with version v1.10.4.</p>
Unity XR	The new version of Unity XR. See the dedicated section below.

If you have a newer version of the plug-ins and see any issues, please report them to us. The best place to do this is in our Unity forum.

Once the Oculus or Vive packages have been installed and you have restarted Unity, by selecting the appropriate “Input Mode” you should be able to see the 3rd party version number and the “Input Type” for each axis or button.



To enable the older VR support in Unity, it must be enabled in the Unity Player Settings under “XR”. XR is an umbrella term, encompassing Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) applications. For more information see:

<https://docs.unity3d.com/Manual/XR.html>

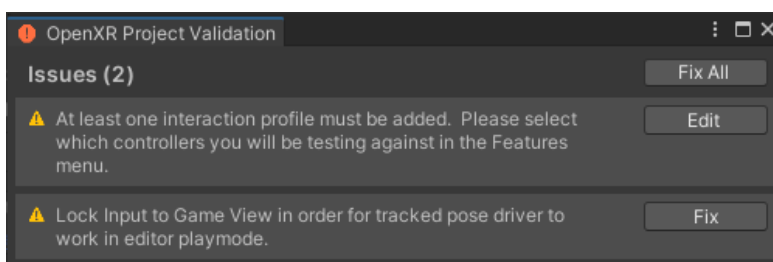
Unity XR - Overview

From Unity 2019 LTS, a new framework is available for XR - Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR). SSC has added support for this in Unity 2020.3 LTS or newer. This system uses a XR Plug-in Framework and is built on top of a Unity XR SDK.

The action-based system uses the “new” Unity Input System which is automatically installed during the basic setup steps provided below.

Here are the basic setup steps if using the Oculus Quest 2 headset:

- 1) In “Build Settings” ensure you have switched to the Android platform and “Texture Compression” is ASTC (the Oculus Quest 2 is an Android device).
- 2) Project Settings->XR Plugin Management
- 3) Install XR Plugin Management
- 4) Tick Open XR (Oculus, Windows Mixed Reality, Unity Mock HMD etc)
- 5) Click "Yes" to enable the Input System when prompted (after a few moments the editor will restart)
- 6) If there is a small yellow warning beside "OpenXR", click it.



- 7) Click "Fix" next to Lock Input to Game View. Leave the other one for now.
- 8) Project Settings, XR Plug-in Management->OpenXR->Android tab

- a. Interaction Profiles, "+", Oculus Touch Controller Profile.
 - b. Feature Groups, Oculus Quest Support
 - c. Play Mode OpenXR Runtime: Oculus (Standalone tab only)
- 9) On the player SSC ship in the scene, change the Input Mode to "Unity XR". For testing, you might want to use Demos\VR\Prefabs\SSCHawk2 (Arcade) VR or SSCInterceptor (Arcade) VR.
- 10) Using the Unity Input System, create an Input Action Asset scriptableobject in the Project pane. TIP: as a shortcut, install the XR Interactive Toolkit and make a copy of the sample "XRI Default Input Actions" scriptableobject which can be installed with XRI. If the XRI RightHand "Move" action doesn't have a binding, add one – see the XRI LeftHand "Move" as an example. If you don't want XRI in your project, you can install it in another Unity project and just import the single asset into your working project. You will also need to manually copy FallbackComposite.cs from the XR Interaction Toolkit, Runtime, Inputs, Composites package cache to your project (right-click on file and select Show in Explorer to copy the file).
- 11) In PlayerInputModule, assign the new Input Action Asset to the slot provided.
- 12) In PlayerInputModule, select "New" next to "XR Camera"
- 13) Do the same for Left and Right Hands.
- 14) On the XR Camera, find the "Tracked Pose Driver", for "Position Action" click "+" and "Add Binding" and set the Path to "<XRHMD>/centerEyePosition" (without the quotes)
- 15) On the XR Camera, find the "Tracked Pose Driver", for "Rotation Action" click "+" and "Add Binding" and set the Path to "<XRHMD>/centerEyeRotation" (without the quotes)
- 16) On the XR Left Hand "Tracked Pose Driver", for "Position Action" click "+" and "Add Binding" and set the Path to "<XRController>{LeftHand}/devicePosition" (without the quotes)
- 17) On the XR Left Hand "Tracked Pose Driver", for "Rotation Action" click "+" and "Add Binding" and set the Path to "<XRController>{LeftHand}/deviceRotation" (without the quotes)
- 18) On the XR Right Hand, in a similar way, configure the "Position Action" and "Rotation Action".

Suggested setup using the modified XRI default actions.

SSC Action	Action Map	Action	Action Type	Data Slot	
Vertical					
Horizontal					
Longitudinal	XRI RightHand	Move	Value	Slot2	
Pitch	XRI LeftHand	Move	Value	Slot2	
Yaw	XRI LeftHand	Move	Value	Slot1	
Roll	XRI RightHand	Move	Value	Slot1	
Primary Fire	XRI RightHand	Activate	Button	Slot1	
Secondary Fire	XRI LeftHand	Activate *	Button	Slot1	
Docking					

* Alternatively, use the XRI RightHand "Select" which is the grip trigger rather than the primary trigger on the Oculus Quest 2.

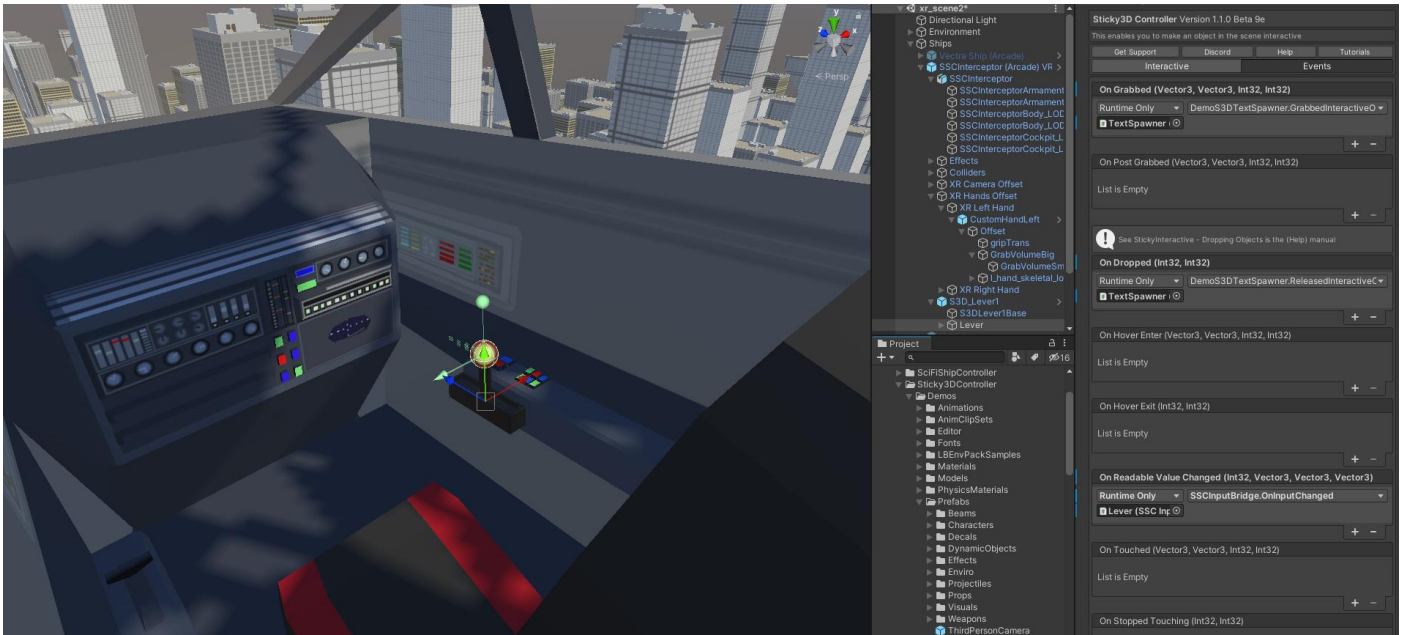
If you want to animate your hands, and also own Sticky3D Controller, you can set that up using the Sticky3D Interactor Bridge component. See the Sticky3D manual for details.

Unity XR – Levers and Joysticks with Sticky3D

The SSC Player Input Module can be configured to handle input from interactive-enabled in-game levers and joystick from Sticky3D Controller (S3D). This should be done AFTER following the instructions in the above (Unity XR – Overview) chapter.

This other asset, which is available from the [Unity Asset Store](#), includes special bridging software to allow SSC VR hands to interact with objects from Sticky3D. These levers and joysticks can be used to provide input to fly your ship in VR. How cool is that?

These tools come with Sticky3D Controller version 1.1.0 or newer. Be sure to check out the “Sticky Interactor Bridge” and “SSC Input Bridge” chapters in the Sticky3D manual for all the details.

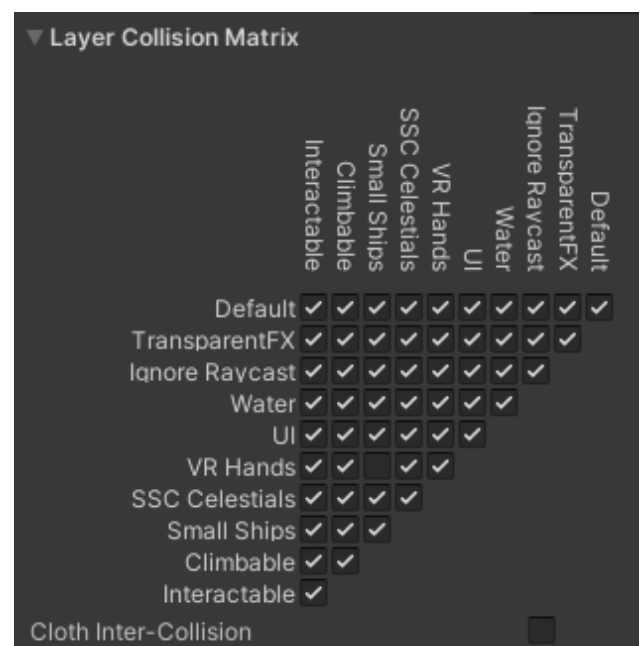


Unity XR – Hands and Physics Collisions

Typically, you don't want your XR hands to collide with the ship itself. If the hands have colliders and rigidbodies, they will push the ship while it's flying. That's not what you want.

Essentially, this is what you need to do:

1. Place your ships that you fly in VR into a separate Unity Layer (say “Small Ships”). On the prefab, change the “Layer” in the Unity Editor to “Small Ships” and click “Yes, change children”.
2. If you don't have one already, create a new Unity “Layer” in the Unity Editor called say “VR Hands”.
3. Find “XR Hands Offset” which should be a child of the ship, and change the Layer to “VR Hands”. Again click “Yes, change children” when prompted.
4. Find any interactive objects under the ship that you do want the hands to touch and use, and set those to say the “Default” Layer. Again, click “Yes, change children” on those objects too. For in-game levers and joysticks, you might want to added them to your “VR Hands” layer to avoid any false collision issues with the ship.
5. As a final step, in the Unity Editor, go to “Project Settings...” and find “Physics”. Now uncheck the intersection between “Small Ships” and “VR Hands”. This will allow your XR hands to interact with



everything in your scene, and your ship, except the actual ship itself.

On the hand rigidbodies, assuming they are Kinematic, set the “Collision Detection” to “Continuous Speculative”. Do the same for your interactive levers and joysticks.

Custom Player Input

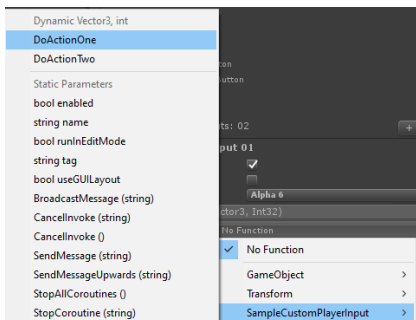
Each of the supported Input Modes (Direct Keyboard, Legacy Unity, Unity Input System, Unity XR, Oculus API, Rewired, and Vive), can take input from their various devices, and push that data to your own custom C# method.

Example of how you might use this feature include:

- Call your own code when the player presses a button on a controller
- Call your own menu code
- Perform a custom action like change the camera position
- Get the value from say a controller trigger
- Activate your own tractor beam
- Modify a ship variable or setting at runtime

To use this in your own game code:

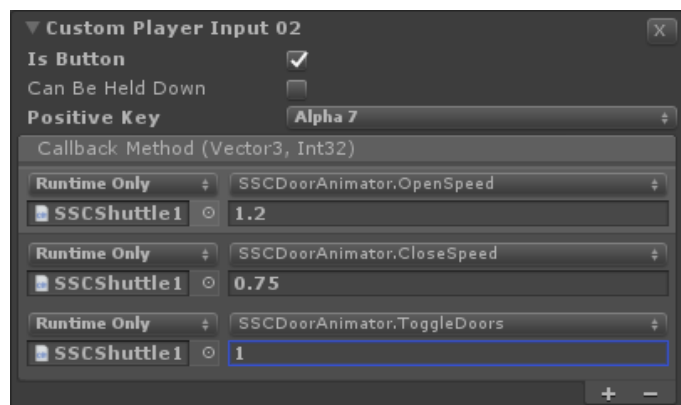
1. Create a new C# public method in your own game code (ensure it takes a Vector3 and int parameter if you want to get the input value and the input type) **
2. In Player Input Module Inspector, add a “Custom Player Input”
3. Select the button or controller input. If you are using Legacy Unity, Unity Input System, or Rewired you may need to configure this in the current input system first.
4. In Player Input Module, under the new Custom Player Input, add a Callback Method event.
5. Drag the gameobject from the scene that includes your game code script onto the empty Object field.
6. Configure the event function by selecting the method you created in item #1 above which should be under “Dynamic Vector3 int” in the popup menu.



For examples, see Demos\Scripts\SampleChangeCameraView.cs or SampleCustomPlayerInput.cs.

** You can also use “static parameters” or just call a method that takes no parameters. For example, you may wish to simply perform an action when the player presses a button on a controller or the keyboard. Or you may wish to always pass a certain parameter to a method when the user presses a button. Static Parameters only support a single parameter.

On the right, is an example of setting the door opening and closing speeds before toggling the door(s) open or closed.



Using Sony Dual Shock 4 with SSC on PC

If you don't have an Xbox One controller, but have a Sony Dual Shock 4, you are all set. You can use the legacy Unity Input method or, if you have the excellent Rewired asset, then the Sony controller will work with the suggestion config here.

To connect your Sony controller to a Windows 10 PC, follow these steps:

Wired

Use a micro-USB to USB cable, plug the small end into the controller and the regular USB into your PC. Your controller is now ready! How easy is that?!

Wireless

1. Make sure the Sony Dual Shock 4 controller is fully charged
2. Turn the controller off
3. On the Windows 10 PC make sure Bluetooth is enabled
4. On Windows 10 PC type "Bluetooth and other device settings" in search bar (bottom left of screen)
5. Click "Add Bluetooth or other device"
6. Click "Bluetooth"
7. On Sony controller, press the "PlayStation" button (button between joysticks) and the "Share" button (just to the right of left arrow pad on the top left of the controller).
8. The light bar on the front of the controller should start flashing.
9. When "Wireless Controller" or "Input" appears, click on it. Windows 10 will then automatically pair the controller with the PC.
10. Now, you're good to go!

Overriding Player Input Module in Code

Although you can write your own version of an input module and send input to a ship with `shipControlModule.SendInput(..)`, you may wish to do a combination of both. For example, maybe you wish to have the player control everything except forward and backward motion. This could be achieved by overriding the Longitudinal axis.

In the Player Input Module's editor, you can choose to "Override in Code" each axis. You can also set this value yourself in code. Then in C# code, you could control the speed. This is demonstrated in the `SampleInputOverride.cs` script which is included in the `Demos\Scripts` folder.

Writing your own Player Input code

In the chapter "Runtime and API" there is a "Demo Script" called "SampleSendShipInput" to get you started. The sample script can be found in the `Demos\Scripts` folder.

Ship Camera Module

Camera Overview

This provides basic camera control. So instead of the fixed camera position you can deliver smooth camera movement out of the box by adding this script to an empty gameobject in the scene.

For convenience, a prefab has been included `SCSM\SciFiShipController\Prefabs\Environment` called "PlayerCamera". Drag and drop it into a scene, assign the Target Ship to your Player ship from the scene, and you're ready to test.

Camera Properties

Property	Description
Start on Initialise	Start the camera rendering when it is initialised. You may wish to disable this when you are switching between multiple cameras in your scene.

Property	Description
Enable on Initialise	Enable camera movement when it is initialised and “Start on Initialise” is on.
Target Ship	The target ship for this camera to follow from the scene.
Target Offset Coords	<p>The coordinate system used to interpret the target offset.</p> <ul style="list-style-type: none"> • CameraRotation: The target offset is relative to the rotation of the camera. • TargetRotation: The target offset is relative to the rotation of the target. • TargetRotationFlat: The target offset is relative to the flat rotation of the target. • World: The target offset is relative to the world coordinate system.
Lock Camera Pos	If enabled, the camera will stay at the same position and rotate towards the target. Only available when Camera Rotation Mode is set to “Aim At Target”.
Target Offset	The offset from the target (in local space) for the camera to aim for.
Lock to Target Pos	If enabled, the camera will stay locked to the optimal camera position. To avoid lag, at runtime, the “Update Type” should be LateUpdate. Check “Debug Mode” in the editor at runtime.
Move Speed	How quickly the camera moves towards the optimal camera position.
Offset Damping	If enabled, this enables the camera to dynamically modify the Target Offset based upon the Ship pitch and yaw inputs. It is only available if Lock to Target Pos is NOT enabled.
Max Pitch Offset Up	The damping maximum pitch Target Offset Up (y-axis)
Max Pitch Offset Down	The damping maximum pitch Target Offset Down (y-axis). What is the lowest Target Offset Y value can be?
Damping Pitch Rate	The rate at which Target Offset Y is modified by ship pitch input. Higher values are more responsive.
Damping Pitch Gravity	The rate at which the Target Offset Y returns to normal when there is no ship pitch input. Higher values are more responsive.
Max Yaw Offset Left	The damping maximum yaw Target Offset left (x-axis)
Max Yaw Offset Right	The damping maximum yaw Target Offset right (x-axis). What is the lowest Target Offset X value can be?
Damping Yaw Rate	The rate at which Target Offset X is modified by ship yaw input. Higher values are more responsive.
Damping Yaw Gravity	The rate at which the Target Offset X returns to normal when there is no ship yaw input. Higher values are more responsive.
Lock Target to Rot	If enabled, the camera will stay locked to the optimal camera rotation. To avoid lag, at runtime, the “Update Type” should be LateUpdate. Check “Debug Mode” in the editor at runtime.
Turn Speed	How quickly the camera turns towards the optimal camera rotation.
Camera Rotation Mode	<p>How the camera rotation is determined.</p> <ul style="list-style-type: none"> • FollowVelocity: The camera rotates to face in the direction the ship is moving in. • FollowTargetRotation: The camera rotates to face the direction the ship is facing in. • AimAtTarget: The camera rotates to face towards the ship itself. • TopDownFollowVelocity: The camera faces downwards and rotates so that the top of the screen is in the direction the ship is moving in.

Property	Description
	<ul style="list-style-type: none"> • TopDownFollowTargetRotation: The camera faces downwards and rotates so that the top of the screen is in the direction the ship is facing in. • Fixed: the camera rotation is fixed
Follow Velocity Threshold	Below this velocity (in metres per second) the forwards direction of the target will be followed instead of the velocity.
Rotation	The target rotation of the camera in world space. Only available when "Camera Rotation Mode" is "Fixed".
Orient Upwards	If enabled, the camera will orient with respect to the world up direction rather than the target's up direction. Off by default in SSC 1.2.2 or newer. This is more useful if the ship is unlikely to be flying straight up or straight down (at which point the camera will "flip").
Update Type	<p>When the camera position/rotation is updated.</p> <ul style="list-style-type: none"> • FixedUpdate: The update occurs during FixedUpdate. Recommended for rigidbodies with Interpolation set to None. • LateUpdate: The update occurs during LateUpdate. Recommended for rigidbodies with Interpolation set to Interpolate. • Automatic: When the update occurs is automatically determined.
Max Shake Strength	The maximum strength of the camera shake. When this is 0, the feature is disabled. Smaller numbers are typically better. Start with something like 0.05.
Max Shake Duration	The maximum duration (in seconds) the camera will shake per incident. A small duration like 0.2 is a good starting point.
Clip Objects	Adjust the camera position to attempt to avoid the camera flying through objects between the ship and the camera. If you don't need this, keep it turned off to avoid unnecessary performance overhead.
Minimum Move Speed	The minimum speed the camera will move to avoid flying through objects between the ship and the camera. High values make clipping more effective. Lower values will make it smoother. Currently this has no effect if Lock to Target Position is enabled.
Minimum Distance	When Clip Objects is enabled, the minimum distance the camera can be from the Ship (target) position. Typically, this is the spheric radius of the ship. If the ship has colliders that do not overlay the target position or centre of the ship gameobject, this value should be set, else set to 0 to improve performance.
Minimum Offset X	The minimum offset on the x-axis, in metres, the camera can be from the Ship (target) when object clipping. This should be less than or equal to the Target Offset X value.
Minimum Offset Y	The minimum offset on the y-axis, in metres, the camera can be from the Ship (target) when object clipping. This should be less than or equal to the Target Offset Y value.
Clip Object Layers	Clip objects in the selected Unity Layers. It cannot be set to "Nothing".
Zoom Duration	The time, in seconds, to zoom fully in or out.
Unzoom Delay	The delay, in seconds, before zoom starts to return to the non-zoomed position.
Unzoomed FoV	The camera field-of-view when no zoom is applied [Default 60]
Zoomed In FoV	The camera field-of-view when fully zoomed in [Default 10]
Zoomed Out FoV	The camera field-of-view when fully zoomed out [Default 90]

Property	Description
Zoom Damping	The amount of damping applied when starting or stopping camera zoom.

The Camera Shake feature will take input from the Target Ship. The actual strength and duration will depend on how much collision or normal damage the ship receives in any one incident. If an incident happens before the current camera shake ends, the shaking will be extended by the new duration.

Top-down Setup

SSC supports a number of popular top-down camera configurations. Although it is not a requirement, a good starting point is to set your ship in 2.5D mode first.

On your player Ship Control Module in the scene, go to the “Control” tab and set the “Input Control Axis” to “Y”. Adjust the “Input Control Limit”, “Input Moving Rigidness”, and “Input Turning Rigidness” as required. See the “Ship Control Module” chapter for more details.

Back on your Ship Camera Module:

1. Set the “Target Offset Coords” to “Target Rotation Flat”.
2. Set the “Target Offset”. E.g., X = 0, Y = 50, Z = 0
3. Select a “Camera Rotation Mode”. E.g., Fixed with “Rotation” set to X = 90, Y = 0, Z = 0.
4. Give it a try.
5. Now change the “Camera Rotation Mode” to “Top Down Follow Velocity” and set the “Follow Velocity Threshold” to 10.
6. Give that a try
7. Now change the “Camera Rotation Mode” to “Top Down Follow Target Rotation”
8. Try that.

Ship Camera Settings ScriptableObject

Optionally, you can create ShipCameraSettings ScriptableObjects in the Assets folder of your project. These can be referenced in your game code. At runtime, in your C# code call shipCameraModule.ApplyCameraSettings(..).

1. Select or create a folder in your game under Assets in your Project panel. Be careful not to use any of the SCSM subfolders.
2. Right-click the folder and select Create, Sci-Fi Ship Controller, Ship Camera Settings.
3. Give it a meaningful name. e.g., DropshipCockpitView

Projectile Module

Every Weapon that fires a projectile requires a prefab with a Projectile Module script attached. For details on using projectile weapons on a Ship, see the “Combat Tab” under the “Ship Control Module” section in this manual.

Projectiles are also used with the Surface Turret Module.

How to Create Projectile Prefabs

To create a new Projectile:

1. In a scene, create an empty gameobject
2. Rename the empty gameobject. E.g., MyProjectile1
3. Either add a mesh and mesh renderer directly to the gameobject, or as a child gameobject
4. Remove any colliders that may have been added in step 3 (SSC doesn’t depend on projectile colliders for projectile collisions)
5. Add a Projectile Module script to the parent gameobject
6. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
7. Reset the prefab parent transform position and rotation to 0,0,0
8. Delete the gameobject from the scene

The Projectile Module script has three options for spawning projectiles from a weapon's canon or fire position.

Projectile system	Description
Standard	The standard behaviour doesn't have pooling or DOTS enabled. It relies on the standard Unity object instantiation and destruction methods. It works well when there are only a few projectiles in the scene.
Pooling	An object pooling system to manage create, re-use, and destroy projectiles. This is more efficient than the "standard" projectile system.
DOTS	This is a high-performance method of creating and destroying projectiles that utilises the new Unity Data-Orientated Technology Stack (DOTS). Projectiles are Entities rather than GameObjects. SSC converts the projectile prefabs to entities at runtime. For games that require high-performance, low overhead projectiles, this is the preferred choice.

How to Setup Projectiles to use DOTS/ECS

We were early adopters of DOTS (from Unity 2019.1). Due to the rapid changing nature of DOTS, we may not support it with your Unity Editor version. We maintained support in U2019.x, U2020.3, and U2021.3.

We will endeavour to fix all the breaking changes Unity introduced with Entities 1.0. Unity completely removed a core feature we use (Hybrid Renderer) with the release of "Entities Graphics". However, at this stage, there is no workaround for this significant breaking change. If you wish to use DOTS projectiles in U2022+ ping us on our Discord channel.

Unity 2020.3 LTS (Entities 0.17.0 and Hybrid Render 0.11.0. Hybrid Render 0.51 only supports URP and HDRP)

1. Open Project Settings, and select Package Manager
2. Under "Advanced Settings" turn on "Enable Preview Packages" and "Show Dependencies"
3. Close Project Settings, and open the Package Manager
4. Change to Packages to "Unity Register"
5. Install the following packages (or newer)
 - Burst 1.4.6
6. "Add package from git URL..." under the + menu at the top left of the package manager.
7. Enter "com.unity.rendering.hybrid" and click "Add" (Hybrid Renderer 0.11.0-preview.42, Entities 0.17.0-preview.41, and Jobs 0.8.0-preview.23 or should be installed)
8. On the projectile prefab, in the Projectile Module script inspector, enable "Use DOTS"

Unity 2021.3.6+ LTS (URP and HDRP only)

1. Open Project Settings, and select Package Manager
2. Under "Advanced Settings" turn on "Enable Pre-Release Packages" and "Show Dependencies"
3. Close Project Settings, and open the Package Manager
4. Change to Packages to "Unity Register"
5. "Add package from git URL..." under the + menu at the top left of the package manager.
6. Enter "com.unity.rendering.hybrid" and click "Add" (Burst 1.6.6, Collections 1.3.1, Hybrid Renderer 0.51.0-preview.32, Entities 0.51.0-preview.32, Mathematics 1.2.6, Platforms 0.51.0-preview.31 and Jobs 0.51.0-preview.32 or newer should be installed)
7. On the projectile prefab, in the Projectile Module script inspector, enable "Use DOTS"

Unity 2022.2.1+ LTS (URP and HDRP only)

1. Open Project Settings, and select Package Manager
2. Under "Advanced Settings" turn on "Enable Pre-Release Packages"

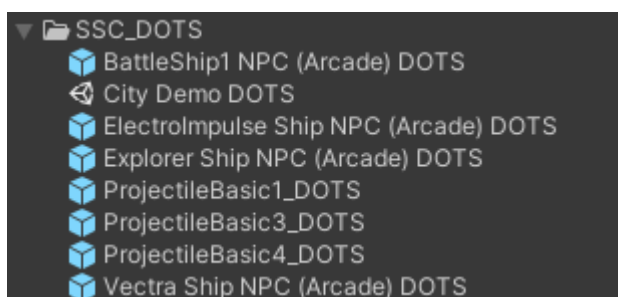
3. Close Project Settings, and open the Package Manager
4. Change to Packages to “Unity Register”
5. Select “Entities Graphics” 1.0.0-pre.15 (or newer) and click “Install” (this will also install Entities 1.0.0-pre.15 or newer)
6. On the projectile prefab, in the Projectile Module script inspector, enable “Use DOTS”

How to Convert scene to use DOTS projectiles

First setup your project to use DOTS (see steps above), ignoring the last step for the Projectile prefab.

To show how to do this, let’s take an existing demo scene and convert it. Don’t worry, the steps look worse than they are – you will basically just be creating and modifying some prefabs.

1. Create a new folder in Project pane, under “Assets”, called “SSC_DOTS”
2. Open the “City Demo” scene (SCSM\SciFiShipController\Demos\Scenes)
3. Save the scene as “City Demo DOTS” in the “SSC_DOTS” folder
4. The next steps will be repeated for each ship or ship prefab.
5. Locate the next ship in the scene. On the Ship Control Module go to the Combat tab, and locate the Weapons (if any).
6. For each Projectile Weapon, click on the “Projectile Prefab” to locate the prefab in the Project pane.
7. If you have already created a DOTS version for the projectile, replace the existing one with the DOTS version in your SSC_DOTS folder and go to the next weapon or ship.
8. Drag it into the scene “Hierarchy” pane (not the scene view)
9. Rename it. E.g., ProjectileBasic4 could be renamed ProjectileBasic4_DOTS
10. On the Projectile Module, click “Use DOTS”
11. Drag the renamed Projectile into the SSC_DOTS folder in your Project pane (click “Original Prefab” when prompted)
12. Delete the prefab from the scene “Hierarchy”
13. On the Ship Control Module, replace the projectile prefab with the DOTS version from the SSC_DOTS folder.
14. Once you have completed all the ships in your scene, you may then need to do the same steps for prefabs of ships that get instantiated at runtime. For example, in the City Demo scene in the “Hierarchy” under “AI Battle” there is a “DemoControlModule” which instantiates squadrons of ships. Locate that component in the scene “Hierarchy”.
15. Under “Grey Squadron” click on the “NPC Ship Prefab” to locate the “Vector Ship NPC (Arcade)” in the Project pane.
16. Drag the NPC ship into the scene “Hierarchy” and rename it. E.g., “Vector Ship NPC (Arcade) DOTS”
17. On the “Combat” tab, locate the projectile weapons and create/replace any projectile prefabs with DOTS versions (you probably already have a set of DOTS projectiles in your SSC_DOTS folder).
18. From the scene “Hierarchy” drag the renamed ship into the SSC_DOTS folder (click “Original Prefab” when prompted)
19. Delete the ship prefab from the scene
20. On the “DemoControlModule”, under the “Grey Squadron” replace the non-DOTS ship prefab with the DOTS one from your SSC_DOTS folder.
21. Repeat the steps 15-20 for the other squadrons.



To see if projectiles are using DOTS, open the Entity Debugger (Unity editor, Windows, Analysis, Entity Debugger) and run the scene in the editor. You should see Entities being created and destroyed.

Projectile Properties

The table below describes the properties you see in the Inspector. Projectile can also be manipulated at runtime (see the “Runtime and API” section later in the manual). You can also check out “Common Issues – Weapons”.

Property	Description
Start Speed	The starting speed of the projectile when it is launched.
Use Gravity	Whether gravity is applied to the projectile.
Damage Type	The type of damage the projectile does when hitting a ship. The amount of damage dealt to a ship upon collision is dependent on the ship's resistance to this damage type. If the damage type is set to Default, the ship's damage multipliers are ignored i.e., the damage amount is unchanged. See also Ship Control Module, Combat tab, “Damage”.
Damage Amount	The amount of damage the projectile does on collision with a ship or object. NOTE: Non-ship objects need a DamageReceiver component.
Collision Mask	The layer mask used for collision testing for this projectile. Default: Everything
Use DOTS	Use Data-Oriented Technology Stack which uses the Entity Component System and Job System to create and destroy projectiles. Has no effect if Unity 2019.1, ECS, and Jobs is not installed.
Use Pooling	Use the Pooling system to manage create, re-use, and destroy projectiles.
Min Pool Size	When using the Pooling system, this is the number of projectile objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of projectiles permitted in the scene at any one time.
Despawn Time	If the projectile has not collided with something before this time (in seconds), it is automatically despawned or removed from the scene
Guide to Target	Rather than being fire and forget, is this projectile guided to a target with kinematics? NOTE: Not currently supported by DOTS.
Guided Max Turn Speed	The maximum turning speed in degrees per second for a guided projectile.
Effects Object	The particle system and/or sound effect prefab that will be instantiated when the projectile hits something and is destroyed. This does not fire when the projectile is automatically despawned. See also Effects Module.
Shield Effects Object	The particle and/or sound effect prefab that will be instantiated, instead of the regular Effects Object, when the projectile hits a shielded ship. This does not fire when the projectile is automatically despawned. See also Effects Module.
Muzzle FX Object	The particle and/or sound effect prefab that will be instantiated when the projectile is fired from a weapon. For muzzle FX objects to move with the weapon fire point, the Effects Module must be poolable and have Is Reparented enabled.
Muzzle FX Offset	The distance in local space that the muzzle Effects Object should be instantiated from the weapon firing point. Typically, only the z-axis would be used when the projectile is instantiated in front or forwards from the actual weapon.

When “Use Pooling” is enabled, the optimal Min/Max Pool Size can be determined by:

1. Running the game in the editor
2. Pausing the game, then finding SSCManager in the scene hierarchy
3. Enabling “Debug Mode”
4. Taking note of the current pool sizes for Projectile Templates

If the current pool size never approaches the Max Pool Size, then the Max Pool Size can be reduced.

Important

When modifying existing prefabs that come with SSC, we recommend creating a new prefab (i.e., a copy of the prefab) and placing it in your own project folder outside the SciFiShipController folder. This way, when you download an SSC update, your changes won’t be overwritten.

How to Use Guided Projectiles

Guide projectiles use a regular Projectile Module as described above. Simply create a new Prefab or create a duplicate (Original) prefab from an existing one and store it your own folder. Ensure “Guide to Target” is enabled.

You then need something to tell the projectile what to target. If you are using a Surface Turret Module or a Turret weapon on a ship, you could use the Auto Targeting Module. On a ship, you need to make sure “Auto Targeting” option is enabled on the weapon.

By default, guided projectiles use Augmented Proportional Navigation to chase a target. If you don’t like our guided projectile algorithm, in code you could tell SSC to use your amazing algorithm! Check out “Projectile API Call Backs” in the “Runtime and API” section.

Customising Projectile Behaviour

If the current options don’t achieve what you want in your game, you can create custom projectile behaviour by creating a custom class that inherits from the ProjectileModule class. This enables you to write custom code in override methods. See Demos\Scripts\SamplePorjectileModule.cs for more details.

Beam Module

Every Weapon that fires a beam requires a prefab with a Beam Module script attached. For details on using beam weapons on a Ship, see the “Combat Tab” under the “Ship Control Module” section in this manual. For Surface Turrets, see the “Surface Turret – Weapon Settings” section of the “Surface Turret Module” chapter.

How to Create Beam Prefabs

To create a new Beam prefab:

1. In a scene, create an empty gameobject
2. Rename the empty gameobject. E.g., MyBeam1
3. Add a Line Renderer directly as a child gameobject
4. On the Line Renderer, do not enable “Use World Space” or “Loop” and we recommend turning off shadows.
5. Ensure the Line Renderer has exactly 2 Positions. Leave them as their default values of 0,0,0
6. Add a Beam Module script to the parent gameobject
7. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
8. Reset the prefab parent transform position and rotation to 0,0,0
9. Delete the gameobject from the scene

The Beam Module script has two options for spawning beams from a weapon’s canon or fire position.

Projectile system	Description
Standard	The standard behaviour doesn’t have pooling. It relies on the standard Unity object instantiation and destruction methods. It works well when there are only a few beams in the

Projectile system	Description
	scene. You may wish to use this if you have also attached your own scripts to the prefab to run some custom code.
Pooling	An object pooling system to manage create, re-use, and destroy beams. This is more efficient than the “standard” beam system. This is enabled by default.

Beam Properties

The table below describes the properties you see in the Inspector. Beams can also be manipulated at runtime (see the “Runtime and API” section later in the manual). You can also check out “Common Issues – Weapons”.

Property	Description
Damage Type	The type of damage the beam does when hitting a ship. The amount of damage dealt to a ship when hit is dependent on the ship's resistance to this damage type. If the damage type is set to Default, the ship's damage multipliers are ignored i.e., the damage amount is unchanged. See also Ship Control Module, Combat Tab, Damage.
Damage Rate	The amount of damage this beam does, per second, to the ship or object it hits. NOTE: Non-ship objects need a DamageReceiver component.
Use Pooling	Use the Pooling system to manage create, re-use, and destroy beams.
Min Pool Size	When using the Pooling system, this is the number of beam objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of beams permitted in the scene at any one time.
Beam Width	The width (in metres) of the beam on the local x-axis.
Min Burst Duration	The minimum amount of time, in seconds, the beam must be active.
Max Burst Duration	The maximum amount of time, in seconds, the beam can be active in a single burst.
Discharge Duration	The time (in seconds) it takes a single beam to discharge the beam weapon from full charge.
Effects Object	The particle system and/or sound effect prefab that will be instantiated when the beam hits something. For Beam Effects, this would typically be a looping effect with a Despawn Time greater than the Max Burst Duration of the Beam. See also Effects Module.
Muzzle FX Object	The particle and/or sound effect prefab that will be instantiated when the beam is fired from a weapon. For muzzle FX objects to move with the weapon fire point, the Effects Module must be poolable and have Is Reparented enabled.
Muzzle FX Offset	The distance in local space that the muzzle Effects Object should be instantiated from the weapon firing point. Typically, only the z-axis would be used when the beam is instantiated in front or forwards from the actual weapon.

Destruct Module

The destruct module uses a pre-created prefab of mesh fragments to produce a destruction effect. It enables you to manage how it behaves as the prefab breaks into fragments.

How to Create Destruct Prefabs

A Destruct prefab has a parent gameobject which contains the “Destruct Module” script and has a position and rotation reset to 0, 0, 0 and a scale of 1,1,1.

Child objects should be the mesh fragments including any materials assigned to the mesh renderers. Typically, you will make the fragments in 3D modelling software from the original model that you want to destruct. If you model was made in a 3D modelling package like Blender, you could use the Object: Cell Fracture feature to create your fragments.

We recommend keeping the number of fragments to a minimum to reduce the performance impact in your game.

You can either add rigidbody components and colliders to each fragment, or you can have the Destruct Module do it for you at runtime. The rigidbody must be on the same gameobject as the mesh renderer for each fragment (that is, it cannot be a child of the fragment within the prefab).

If you do not want to use Mesh Colliders (which have a higher performance overhead), you should add the colliders to the fragments yourself in the prefab, and disable “Add Mesh Colliders” in the Destruct Module Inspector.

Destruct Properties

The table below describes the properties you see in the Inspector. Destruct prefabs can also be manipulated at runtime (see the “Runtime and API” section later in the manual). You can also check out “Common Issues – Destruct prefabs”.

Property	Description
Explode on Start	Should the explosion occur immediately the scene is started or the module is instantiated? This should be disabled if used with a pooling system.
Use Pooling	Use the Pooling system to manage create, re-use, and destroy destruct prefabs.
Min Pool Size	When using the Pooling system, this is the number of destruct objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of destruct objects permitted in the scene at any one time.
Start Static	Start in Static mode rather than Dynamic. This is not available when “Disable Rigidbody Mode” is set to “Destroy”.
Add Rigid Bodies	Add rigidbodies to the fragments in the prefab
Add Mesh Colliders	Add mesh colliders to the fragments in the prefab
Explosion Radius	The default effective range, in metres, of the blast
Explosion Power	The default power of the blast
Unmoving Velocity	When the speed (in metres per second) in any direction of the fragment falls below this value, the fragment is considered to have stopped moving
Max Time Unmoving	If a fragment has been unmoving for more than the maximum time, set the object as static and “disable” the rigidbody according to the “Disable Rigidbody Mode”.
Total Mass	The total mass of all fragments in the prefab
Calc Mass by Bounds	This may be more accurate when there the is a lot of variation between the size of each fragment. This method is slower during the initial configuration phase and may affect performance of non-pooled modules.
Drag	The amount of drag the fragments have. A solid block of metal would be 0.001, while a feather would be 10.
Angular Drag	The amount of angular drag the fragments have

Property	Description
Use Gravity	Fragments are affected by gravity
Interpolation	The rigidbody interpolation
Collision Detection	The rigidbody collision detection mode
Disable Rigidbody Mode	How the rigidbodies are disabled when they are considered to be static. "Destroy" removes the rigidbody component and adds it back in as needed - best performance for unmoving objects. "Set As Kinematic" sets the rigidbody to kinematic - half/half performance "Don't Disable" uses Unity default rigidbody behaviour where rigidbodies go into sleep mode until something collides with them – best for objects being moved "Destroy" is best for objects far away from each other, and "Don't Disable" is best for objects close to each other
Despawn Condition	The conditions under which this object despawns. Options include "Time" and "Don't Despawn".
Despawn Time	After this time (in seconds), the destruct object is automatically despawned or removed from the scene when the "Despawn Condition" is "Time".
Wait Until not Rendered	Wait until the fragment is not being rendered by the camera before being despawned
Wait Until Static	Wait until the fragment is set to static before being despawned

Effects Module

This module enables you to implement effects behaviour on the object it is attached to. This can include multiple child particle systems and/or an audio source attached to the parent gameobject. Create a prefab so that it can be assigned within the Ship Control Module or Projectile Module.

When "Use Pooling" is enabled, and an Audio Source is attached to the prefab's parent gameobject, ensure that "Play On Awake" is not enabled on the Audio Source.

Property	Description
Use Pooling	Use the Pooling system to manage create, re-use, and destroy effects objects.
Min Pool Size	When using the Pooling system, this is the number of effects objects kept in reserve for spawning and despawning.
Max Pool Size	When using the Pooling system, this is the maximum number of effects objects permitted in the scene at any one time.
Despawn Time	After this time (in seconds), the effects object is automatically despawned or removed from the scene.
Is Reparented	Does this object get parented to another object when activated? If so, it will be reparented to the pool transform after use. This should be enabled on poolable effects used for Muzzle FX on a projectile or beam weapon.
Default Volume	If an audio source is included, the volume can be optionally set at runtime in C# code to the default volume.

When "Use Pooling" is enabled, the optimal Min/Max Pool Size can be determined by:

1. Running the game in the editor
2. Pausing the game, then finding SSCManager in the scene hierarchy
3. Enabling “Debug Mode”
4. Taking note of the current pool sizes for Effects Templates

If the current pool size never approaches the Max Pool Size, then the Max Pool Size can be reduced.

SSC Manager

The Manager is automatically added to the scene if it doesn't already exist at runtime. It includes beam, projectile, and effects object management for all ships along with Location and Path management. Each scene should only have one SSC Manager. However, we do support you loading (and unloading) additive scenes where each scene would have it's own SSC Manager. This might be done in code using the Unity SceneManager API.

The Manager includes the following core functionality:

- Beam, Projectile, and Effects object management
- Location and/or Path in-Editor editing
- Location and Path runtime API
- Runtime in-Editor Beam, Projectile, and/or Effects Object pool size estimation

Although Beam, Projectile, and Effects object management is mostly for internal use only, the other core functionality of the Manager is aimed at developers like you.

Adding SSC Manager to a Scene

Some features require the SSC Manager to be present in the scene in the editor. To add it to the scene:

1. go to the scene Hierarchy
2. click “Create” or “+” depending on your Unity version
3. 3D Object->Sci-Fi Ship Controller
4. SSC Manager

Locations and Paths - Overview

Sci-Fi Ship Controller comes with its own Location and Path editor. Locations and Paths may be configured in the Unity Editor by selecting SSC Manager in the scene Hierarchy panel and/or by using the API methods provided. A Location is a world space position without a game object which can be used by ships in your game.

Location examples:

- Enemy Base
- Friendly Base
- Target location
- Route marker
- Ship spawn point or destination

Paths consist of zero, one, or more Location slots. The slots can be occupied by a Location or a dummy Location, the latter being considered an unassigned Location. A Location can be part of zero, one or more Paths.

When a Location is part of a Path, the Location slot records in and out Control points or tangents. These determine the shape and length of the Path section between Locations.

Locations and Paths can be used with your own AI system or in general gameplay. They are fully supported with our Ship AI system.

Locations and Paths - Creating

There are a number of ways a Location can be added to the scene using the Unity Editor after the SSC Manager has been added to the scene.

- From the Location tab, set focus to the scene, hover the mouse pointer over the desired position, and press the “+” key (don’t click the mouse)
- From the Location tab, in the inspector click the “+” button to add a Location to the end of the list. By default, it will be placed at 0,0,0. Type “new” in the Filter, then the (F)ind button to show it in the scene.
- From the Location tab, in the inspector, click the (I)nsert button next to an existing Location. In the scene view move the selected duplicate Location to the desired position.
- From the Path tab, add a new Path, or (A)ctivate an existing Path. Set focus to the scene, hover the mouse pointer over the desired position, and press the “+” key (don’t click the mouse). A new Location will be created and appended to the Active Path.

In the scene view, there is also a context-sensitive menu which is available when the Unity Move tool is enabled and the right-mouse button is pressed. If the Path tab in the inspector is selected, the context menu will have extra commands relative to the active Path.

When the “+” key is pressed over the scene view to add a new Location, SSC attempts to place the Location in front of the closest line-of-sight object. The Location will be offset along the mesh normal of that object by the distance you set in the inspector. When adding Locations to Paths, each Path can have a different offset value. If no objects are in the line of sight, the Location is placed in front of the scene view camera.

Editing Paths

To edit an existing Path, select the SSC Manager in the scene Hierarchy. Then on the Paths tab, press the “A” button to activate the Path.

Locations in the Path can be selected by clicking the checkbox in the inspector, or by clicking the Location in the scene view. To multi-select Locations in the scene view hold down the SHIFT key.

The context menu in the scene view can also be used when the Unity Move or Transform tool is enabled to select, unselect and/or delete Locations. When a Location slot is deleted using the inspector, the action Location is not deleted.

To move multiple Locations in the same active Path at the same time, hold down the SHIFT key while moving one of the selected Locations.

There are two Control points at each Location in the active Path. If the Unity Free Hand tool is enabled, the selected Control point can be moved in any direction. If the Unity Move tool is enabled, a XYZ position handle is used to move the Control point.

To change the Y-axis position of selected Locations in a Path, in the inspector, above the list of Path Locations, click the (M)odify attribute button, set the “New Position Y-axis” or “Add Position Y-axis” , and click “Change”.

To reverse the direction of a Path, in the inspector, above the list of Path Locations, click the “<->” button.

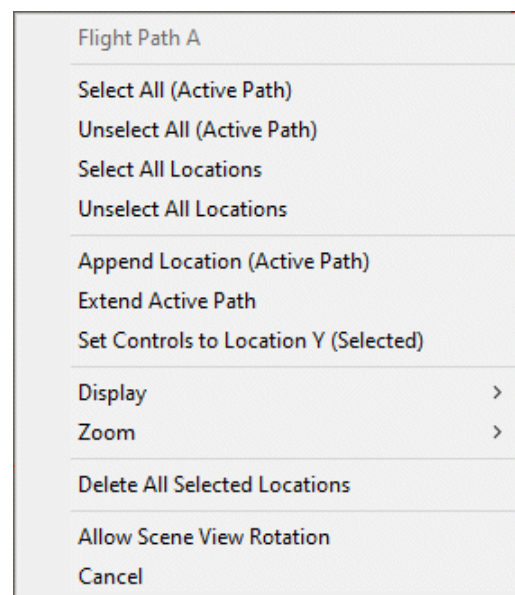
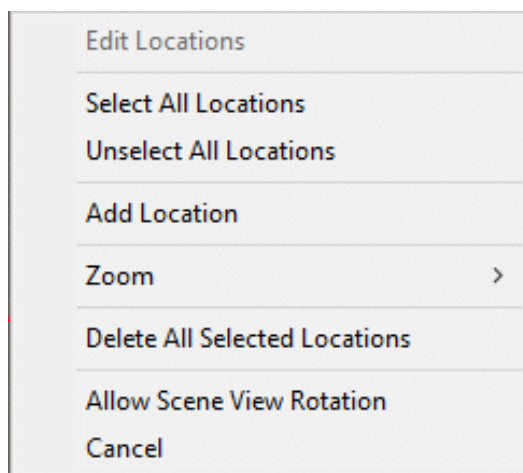
Location Properties

Property	Description
Default Normal Offset	The distance to place a new Location away from the line-of-sight object
Location Name	Name of the location. E.g. Enemy Base, Route Marker 2, Command Post
Location Position	World-space coordinates of the location
Visible to Radar	Is this Location visible to the radar system? If any Location (or ship) has this set, and the Radar system hasn’t been added to scene, the SSC Radar system will be added automatically at runtime.
Radar Blip Size	The relative size of the blip for this Location on the Radar mini-map
Faction Id	The faction or alliance the Location belongs to. This can be used to identify if a Location is friend or foe. Default (neutral) is 0.

Path Properties

Property	Description
Path Name	Name of the Path. Unique names are best.
Total Distance	The total spline length or distance of the Path
Closed Circuit	Is this Path a closed circuit or loop?
Display Number Labels	Display the Location number labels in the scene view
Display Name Labels	Display the Location name labels in the scene view
Display Distances	Display cumulative distances in the scene view
Line Colour	The colour of the Path drawn in the scene.
Default Normal Offset	The distance to place a new Location on the Path away from the line of sight object.

The in-scene context menu will display different items based on which tab is selected in the SSCManager. When Locations are selected, the “Edit Locations” context menu will appear when right-clicking in the scene. If the Path tab is selected *and* a Path is being edited, the Active Path context menu will be displayed. Note, a Path must have been Activated for Editing, before the Path context menu is displayed.



Object Pool Estimation

When using pooling for Beams, Destruct objects, Projectiles and/or Effects Objects, the Manager can be used to estimate the size of the pools required. The optimum sizes can be estimated by:

1. Running the game in the editor
2. Pausing the game, then finding SSCManager in the scene hierarchy
3. Enabling “Debug Mode”
4. Taking note of the current pool sizes for Beam, Destruct, Projectile and/or Effects Templates

For more information see the chapters on Beam Module, Destruct Module, Projectile Module, and Effects Module earlier in this manual.

Ship AI System

Ship AI Overview

For a ship to be controlled by the AI system, it must have a Ship AI Input Module component attached to it. This operates similarly to the player input module – it calculates inputs through the AI system and passes them to the ship using the `SendInput` method.

There are three parts to the AI system: AI states, AI Behaviours and the behaviour combiner.

At any given time, an AI ship is in one (and only one) state. The AI state of a ship should be chosen by what its current goal is. For instance, if a ship needs to move to a specific location in the game world, it should be in the “Move To” state, whose goal is to reach a given position. The AI state is linked with a state method (a function written in code), which converts that goal into a series of behaviours that could be used to achieve that goal. Using the previous example, the aforementioned “Move To” state typically has two main behaviours: Obstacle avoidance and seek. Obstacle avoidance directs ships to take preventative action to avoid collisions, while seek directs the ship to move directly towards a given position. Together, they allow the ship to move towards a specified location (which was the original goal).

The final part of the AI system is the behaviour combiner, which takes the behaviours outputted by the current state and converts them into a final output behaviour. For example, if the behaviour combiner chosen was “Priority Only”, the first behaviour with a nonzero output would be chosen. In the “Move To” example, the obstacle avoidance behaviour can return two types of outputs. If there is an obstacle in front of the ship that it needs to take action to avoid, it will return a nonzero output detailing what action needs to be taken. Otherwise, it will return a zero output, indicating that this behaviour requires no action from the ship. The seek behaviour always returns a nonzero output. So, in this example, if the obstacle avoidance behaviour returns a nonzero output, the final output will be the obstacle avoidance behaviour output; otherwise, the final output will be the seek behaviour output.

Typically, AI states are what you use to control the behaviour of an AI ship. While AI Behaviours are also customisable, the uses cases that require such customisation are far rarer. Currently, Sci-Fi Ship Controller has five states by default: “Idle”, “Move To”, “Dogfight”, “Docking”, and “Strafing Run”. You can also create your own custom AI states if you need functionality different to these default states.

Ship AI Input Module

The Ship AI Input Module is used to define specific characteristics of an AI ship, such as its obstacle avoidance quality and movement algorithm. It is also the module that is used when interacting with the AI system through code.

Property	Description
Initialise on Awake	If enabled, <code>Initialise()</code> will be called as soon as <code>Awake()</code> runs. This should be disabled if you are instantiating the <code>ShipAIInputModule</code> through code.
Enable on Initialise	Will the main update loop perform calculations and send input to the ship as soon as it is initialised? [Default: true]
Movement Algorithm	The algorithm used for calculating AI movement. E.g., Planar Flight, Planar Flight with Banking, or Full 3D Flight.
Obstacle Avoidance Quality	The quality of obstacle avoidance for this AI ship. Lower quality settings will improve performance.
Obstacle Layer Mask	Layermask determining which layers will be detected as obstacles when raycasted against. Exclude layers that you don't want the AI ship to try and avoid using obstacle avoidance.
Raycast Start Offset	If the ship has colliders that do not overlap the centre of the ship, use this with Obstacle Avoidance. This moves the raycasts forward from the centre of the ship to prevent it seeing its own colliders. 0 = OFF

Property	Description
Path Following Quality	The quality of path following for this AI ship. Lower quality settings will improve performance.
Max Speed	The max speed for the ship in metres per second. The ship may not be able to attain this speed due to things like thruster output, and drag etc. This is simply the maximum allowable.
Ship Radius	The supposed radius of the ship (approximated as a sphere) used for obstacle avoidance.
Targeting Accuracy	The accuracy of the ship at shooting at a target. A value of 1 is perfect accuracy, while a value of 0 is the lowest accuracy.
Turn Pitch Threshold	Only use pitch to steer when the ship is within the threshold (in degrees) of the correct yaw/roll angle. Only available for Full 3D Flight.
Roll Bias	When turning, will the ship favour yaw (i.e., turning using yaw then pitching) or roll (i.e., turning using roll then pitching) to achieve the turn? Lower values will favour yaw while higher values will favour roll. Only available for Full 3D Flight.
Max Bank Angle	The maximum bank angle (in degrees) the ship should bank at while turning. Only relevant when movementAlgorithm is set to PlanarFlightBanking.
Max Bank Turn Angle	The turning angle (in degrees) to the target position at which the AI will bank at the maxBankAngle. Lower values will result in the AI banking at a steeper angle for lower turning angles. Only relevant when movementAlgorithm is set to PlanarFlightBanking.
Max Pitch Angle	The maximum pitch angle (in degrees) that the AI is able to use to pitch towards the target position. Only relevant when movementAlgorithm is set to PlanarFlight or PlanarFlightBanking.
Debug Mode	Provides in-editor data that can help determine what an AI ship is doing at runtime. It provides data like the current AIState, speed, and the assigned target (Ship / Path / Location / Position).

AI State Interaction

Setting a Ship's AI State

The state an AI ship is in can be set by calling the SetState method, as shown below:

```
// Set the AI state of shipAIInputModule to the state with ID of newStateID
shipAIInputModule.SetState(newStateID);
```

To set a state, you need to know the state's ID. If it is a default state, you can get the ID using the AIState class. For example, if you wanted to set the state to the Idle state, you would use the following:

```
// Set the AI state of shipAIInputModule to the Idle state
shipAIInputModule.SetState(AIState.idleStateID);
```

If the state is a custom state you have created, you will have received an ID integer when you created the state (see the Creating and Using a Custom State Section).

Getting a Ship's AI State

You can find out what state a ship is currently in by calling the GetState method, as shown below:

```
// Get the current AI State ID of shipAIInputModule
int currentStateID = shipAIInputModule.GetState();
```

Setting State Inputs

Once you have set the AI state for a ship, you typically need to provide inputs for it. The AI state tells the AI ship what its general goal is, but the state inputs give specifics of that goal. For instance, the “Move To” state tells an AI ship that its goal is to move to a particular position, but the TargetPosition input can be used to tell the AI ship which position it should move to.

Each state requires different inputs to operate correctly. To find out which inputs are required for the default states, you can hover over the corresponding stateID variable in your code editor (for example, the AIState.moveToStateID for the “Move To” state).

The target position input is a position in world space that the AI ship should move towards. It can be set by calling the AssignTargetPosition method, as shown below:

```
// Set the target position of shipAIInputModule
shipAIInputModule.AssignTargetPosition(newTargetPosition);
```

The target location input is a location that the AI ship should move towards. It can be set by calling the AssignTargetLocation method, as shown below:

```
// Set the target location of shipAIInputModule
shipAIInputModule.AssignTargetLocation(newTargetLocation);
```

The target path input is a path that the AI ship should follow. It can be set by calling the AssignTargetPath method, as shown below:

```
// Set the target path of shipAIInputModule
shipAIInputModule.AssignTargetPath(newTargetPath);
```

The target rotation input is a rotation in world space that the AI ship should rotate towards. It can be set by calling the AssignTargetRotation method, as shown below:

```
// Set the target rotation of shipAIInputModule
shipAIInputModule.AssignTargetRotation(newTargetRotation);
```

The current target path location index input is the index of the current location in the target path that the AI ship should move towards. It can be set by calling the SetCurrentTargetPathLocationIndex method, as shown below:

```
// Set the new target path location index of shipAIInputModule
shipAIInputModule.SetCurrentTargetPathLocationIndex(newTargetPathLocationIndex);
```

The target ship input is a ship (that the AI ship should typically attack). It can be set by calling the AssignTargetShip method, as shown below:

```
// Set the target ship of shipAIInputModule
shipAIInputModule.AssignTargetPosition(newTargetPosition);
```

The ships to evade input is a list of ships that the AI ship should evade. It can be set by calling the AssignShipsToEvade method, as shown below:

```
// Set the list of ships to evade of shipAIInputModule
shipAIInputModule.AssignShipsToEvade(newShipsToEvadeList);
```

The surface turrets to evade input is a list of surface turrets that the AI ship should evade. It can be set by calling the AssignSurfaceTurretsToEvade method, as shown below:

```
// Set the list of surface turrets to evade of shipAIInputModule
shipAIInputModule.AssignSurfaceTurretsToEvade(newSurfaceTurretsToEvadeList);
```

The target radius input is a distance in metres, typically indicating the size of a target for the AI ship (although it will vary based on the current state). It can be set by calling the AssignTargetRadius method, as shown below:

```
// Set the target radius of shipAIInputModule
shipAIInputModule.AssignTargetRadius(newTargetRadius);
```

The target distance input is a distance in metres, typically indicating the minimum or maximum distance the AI ship should aim to reach from a particular target position. It can be set by calling the AssignTargetDistance method, as shown below:

```
// Set the target distance of shipAIInputModule
shipAIInputModule.AssignTargetDistance(newTargetDistance);
```

The target angular distance input is an angular distance in degrees, typically indicating the minimum or maximum angular distance the AI ship should aim to reach from a particular target rotation. It can be set by calling the AssignTargetAngularDistance method, as shown below:

```
// Set the target angular distance of shipAIInputModule
shipAIInputModule.AssignTargetAngularDistance(newTargetAngularDistance);
```

The target velocity input is a velocity in world space that the AI ship should attempt to match. It can be set by calling the AssignTargetVelocity method, as shown below:

```
// Set the target velocity of shipAIInputModule
shipAIInputModule.AssignTargetVelocity(newTargetVelocity);
```

Getting State Inputs

You can get the current target position by calling the GetTargetPosition method, as shown below:

```
// Get the target position of shipAIInputModule
Vector3 currentTargetPosition = shipAIInputModule.GetTargetPosition();
```

You can get the current target location by calling the GetTargetLocation method, as shown below:

```
// Get the target location of shipAIInputModule
LocationData currentTargetLocation = shipAIInputModule.GetTargetLocation();
```

You can get the current target path by calling the GetTargetPath method, as shown below:

```
// Get the target path of shipAIInputModule
PathData currentTargetPath = shipAIInputModule.GetTargetPath();
```

You can get the current target path location index by calling the GetCurrentTargetPathLocationIndex method, as shown below:

```
// Get the new target path location index of shipAIInputModule
int currentTargetPathLocationIndex = shipAIInputModule.GetCurrentTargetPathLocationIndex();
```

You can get the current target ship by calling the GetTargetShip method, as shown below:

```
// Get the target ship of shipAIInputModule
Ship currentTargetShip = shipAIInputModule.GetTargetShip();
```

You can get the current list of ships to evade by calling the GetShipsToEvade method, as shown below:

```
// Get the list of ships to evade of shipAIInputModule
List<Ship> currentShipsToEvadeList = shipAIInputModule.GetShipsToEvade();
```

You can get the current list of surface turrets to evade by calling the GetSurfaceTurretsToEvade method, as shown below:

```
// Get the list of surface turrets to evade of shipAIInputModule
List<SurfaceTurretModule> currentSurfaceTurretsToEvadeList =
    shipAIInputModule.GetSurfaceTurretsToEvade();
```

You can get the current target radius by calling the `GetTargetRadius` method, as shown below:

```
// Get the current target radius of shipAIInputModule
float currentTargetRadius = shipAIInputModule.GetTargetRadius();
```

You can get the current target distance by calling the `GetTargetDistance` method, as shown below:

```
// Get the current target distance of shipAIInputModule
float currentTargetDistance = shipAIInputModule.GetTargetDistance();
```

You can get the current target angular distance by calling the `GetTargetAngularDistance` method, as shown below:

```
// Get the current target angular distance of shipAIInputModule
float currentTargetAngularDistance = shipAIInputModule.GetTargetAngularDistance();
```

You can get the current target velocity by calling the `GetTargetVelocity` method, as shown below:

```
// Get the current target velocity of shipAIInputModule
Vector3 currentTargetVelocity = shipAIInputModule.GetTargetVelocity();
```

State Completion Status

Certain states can provide a status on whether or not they have been completed. This can be accessed either by checking `HasCompletedStateAction` or by specifying a callback method.

An example of checking `HasCompletedStateAction` is given below:

```
// If the current state action has been completed...
if (shipAIInputModule.HasCompletedStateAction())
{
    // ... do something
}
```

An example of how to specify and declare a callback method is given below:

```
// Specify a callback method: This will be called
// when the current state action has been completed
shipAIInputModule.callbackCompletedStateAction = CompletedStateActionCallback;

/// <summary>
/// The method that will be called when the current state action has been completed.
/// </summary>
private static void CompletedStateActionCallback (ShipAIInputModule shipAIInputModule)
{
    // ... do something here
}
```

The argument of type `ShipAIInputModule` is required; this contains data relating to the AI ship that has just completed its current state action.

State Stage Index

The state stage index is an integer that can be used to keep track of what “stage” has been reached in an AI state. An example usage of this is a state that requires two sub-actions to be completed, one after the other. This could be achieved by making the actions two different states, but if there are a large number of sub-actions this can lead to having an excessive number of states. Instead, the two sub-actions can be made into a single state, with the state stage index being used to store whether or not the first sub-action has been completed. When the state is first entered, the state stage index is set to zero. When the first sub-action is completed, the state stage index could be set to one. Then it is easy to check whether the first sub-action has been completed by checking whether the state stage index is set to zero or one.

You can get the state stage index by calling the `GetCurrentStateStageIndex` method, as shown below:

```
// Get the current state stage index of shipAIInputModule
int currentStateStageIndex = shipAIInputModule.GetCurrentStateStageIndex();
```

You can set the state stage index by calling the `SetCurrentStateStageIndex` method, as shown below:

```
// Set the state stage index of shipAIInputModule
shipAIInputModule.SetCurrentStateStageIndex(newStateStageIndex);
```

Typically, the state stage index should only be set from within a custom state method. As mentioned previously, when `SetState` is called (to enter a new state), the state stage index is set to zero.

Sample AI Scripts

Six sample AI scripts are included in Sci-Fi Ship Controller with examples of the state interaction described above in practice. If you wish to use something similar in your own game, create a new script in your own namespace to avoid it getting overwritten by Sci-Fi Ship Controller updates.

Script name	Description
SampleAttack	An example of how to call a custom method in your game that assigns a target ship to an attacking AI-controlled ship. When a target ship is assigned, the AI ship will be put into the Dogfighting AIState. When no target ship is assigned, the AI ship will attempt to return to its original spawning position (if it has been set).
SampleChase	A simpler example of SampleAttack where a target ship is set and the AI ship is placed in the Dogfight AIState.
SampleEvade	Executes a simple state machine for determining state information for an "evade" AI. An AI ship is assigned a target Path to follow, while attempting to evade up to 5 other ships.
SampleFlyToPosition	This sample shows how you can fly an AI ship to a given position in 3D space
SamplePatrol	Executes a simple state machine for determining which state a "patrol" AI should be in. A ship will follow a given Path unless it encounters an enemy ship which it will attempt to attack. If contact is lost with enemy ship, the AI ship will return to patrolling the Path.
SampleRace	Executes a simple state machine for determining state information for a "race" AI. A ship is assigned a target Path, and placed in the "Move To" AIState.

There are also other scripts that use AI. For example, `DemoFlyToLocation` shows how to write a custom AIState and custom AI Behaviour.

Default AI States

Sci-Fi Ship Controller comes with a number of states by default.

State	Description	Required Inputs	Optional Inputs
Idle	Brings the ship to a stop and prevents it from moving	None	None
Move To	Moves towards either <code>TargetPosition</code> , <code>TargetLocation</code> or <code>TargetPath</code> , depending on which was assigned. If <code>ShipsToEvade</code> is provided, moves out of the targeting regions of (up to five) ships in the list. The state action is set as completed when the ship is within the	<code>TargetPosition</code> OR <code>TargetLocation</code> OR <code>TargetPath</code>	<code>ShipsToEvade</code>

State	Description	Required Inputs	Optional Inputs
	ship radius of TargetPosition / TargetLocation or it if reaches the end of TargetPath.		
Dogfight	Attacks TargetShip, whilst also attempting to evade it if TargetShip ends up behind it. The state action is set as completed when TargetShip is destroyed	TargetShip	None
Docking	Moves directly towards TargetPosition (which is assumed to be moving with velocity TargetVelocity). When it gets within TargetRadius, attempts to match TargetRotation. The state action is set as completed once the ship is within TargetDistance of TargetPosition and TargetAngularDistance of TargetRotation	TargetPosition, TargetRotation, TargetRadius, TargetDistance, TargetAngularDistance, TargetVelocity	None
Strafing Run	Moves directly towards TargetLocation / TargetPosition until it gets within TargetRadius. Then it moves past and away from TargetLocation / TargetPosition until it escapes the TargetRadius, at which point it sets the state action as completed.	TargetPosition OR TargetLocation, TargetRadius	SurfaceTurretsToEvade

Docking and Strafing Run States are currently in Technical Preview.

Custom AI States

Creating and Using a Custom State

All states (including the default states) are global to your scene (although they do not persist between scenes). Hence, custom states are not stored on individual ships but as static members of the AIState class. You can create a custom state by calling the AIState.AddState method, as shown below:

```
// Creates a new state named "New State Name" using NewStateMethod as the
// state method and with the PriorityOnly state combiner
int newStateID = AIState.AddState("New State Name", NewStateMethod,
    AIState.BehaviourCombiner.PriorityOnly);
```

The AIState.AddState method returns the state ID of the new state as an integer. You can then use this state ID for setting the state of an AI ship by calling the SetState method. As arguments, you need to provide a name, the state method to use (see the Writing a Custom State Method section) and the behaviour combiner (see the State Behaviour Combiner section).

Writing a Custom State Method

All states (including the default states) must have a state method. This is the method that will be called each frame by any AI ships in this state. The state method takes input data from a ship and converts it into a series of behaviour inputs that the AI system will use to calculate inputs to send to the ship control module.

A typical declaration of a state method can be seen below:

```

/// <summary>
/// The state method that will be called for any ships in the corresponding state.
/// </summary>
/// <param name="stateMethodParameters"></param>
private static void NewStateMethod (AIStateMethodParameters stateMethodParameters)
{

}

```

The method is usually made static (although it doesn't necessarily have to be) since states are global to the scene. The argument of type `AIStateMethodParameters` is required; this contains data relating to the AI ship and its current state inputs and will be passed in by the AI ship.

Property	Description
AI Behaviour Inputs List	The list of AI behaviour inputs. Modify these behaviour inputs to control the output of the state.
Ship Control Module	The ship control module instance of the AI ship. The ship instance can be obtained using <code>shipControlModule.shipInstance</code> (which can be then be used to obtain most relevant ship data).
Ship AI Input Module	The ship AI input module instance of the AI ship. This can be used to obtain most relevant AI data, and also has a number of useful AI helper functions.
Target Position	The target position input from the AI ship.
Target Location	The target location input from the AI ship.
Target Path	The target path input from the AI ship.
Target Ship	The target ship input from the AI ship.
Ships To Evade	The ships to evade input from the AI ship.

The aim of a state method is to populate the AI behaviour inputs found in the `aiBehaviourInputsList` variable of the `AIStateMethodParameters` object. Each behaviour input that you want to have needs to have two things set: Its behaviour type (what type of behaviour input it is) and its inputs (what inputs will be sent to the behaviour input to calculate its eventual output).

The behaviour type of a behaviour input is how you specify what the behaviour input will do. These correspond to different types of behaviours a ship could have. For example, if the behaviour type is set to "Seek", if that particular behaviour input is selected by the behaviour combiner, the output sent to the ship will tell it to move towards a specified position. The specified position would be given by the inputs sent to the behaviour. Each behaviour has a set of inputs it requires. In the "Seek" example, the required inputs are the target position and the weighting. The aforementioned specified position would be set to the target position. The weighting of a behaviour input is a required input for all behaviour inputs, and for certain behaviour combiners tells the combiners how to combine the behaviour with the other behaviours (for more information, see the State Behaviour Combiner section). However, no matter the behaviour combiner, the weighting must be set to a value greater than zero for the behaviour to be used.

An example of code for a state method can be seen below:

```
// Sets the first behaviour to "Obstacle Avoidance". Obstacle avoidance has
// only one required input, which is weighting. You can see this for
// yourself by hovering over "ObstacleAvoidance" in your code editor, and
// the same for every other behaviour (except for custom behaviours)
stateMethodParameters.aiBehaviourInputsList[0].behaviourType =
    AIBehaviourInput.AIBehaviourType.ObstacleAvoidance;
// Sets the weighting of the first behaviour to 1. Only behaviours that have
// a weighting set to a value other than zero will be used
stateMethodParameters.aiBehaviourInputsList[0].weighting = 1f;

// Sets the second behaviour to "Evasion". Evasion has three required
// inputs: Target position, target velocity and weighting
stateMethodParameters.aiBehaviourInputsList[1].behaviourType =
    AIBehaviourInput.AIBehaviourType.Evasion;
// Sets the target position of the second behaviour to the target ship's position
stateMethodParameters.aiBehaviourInputsList[1].targetPosition =
    stateMethodParameters.targetShip.TransformPosition;
// Sets the target velocity of the second behaviour to the target ship's velocity
stateMethodParameters.aiBehaviourInputsList[1].targetVelocity =
    stateMethodParameters.targetShip.WorldVelocity;
// Sets the weighting of the second behaviour to 1
stateMethodParameters.aiBehaviourInputsList[1].weighting = 1f;
```

The index in the behaviour input list determines the order in which the behaviours will be considered by the behaviour combiner, and hence their priority. In the above example, obstacle avoidance is given the first index (0) and so has the highest priority in this state. Evasion is given the second index (1) and so has the second-highest (which in this example is in fact the lowest) priority in this state.

Since the new state in this example is using the “Priority Only” behaviour combiner, the effect of this new state will be the following:

The first priority is obstacle avoidance, so the AI ship first checks whether it needs to take preventative action to avoid an obstacle:

- If it decides that it does need to take preventative action, it will use the output of the “Obstacle Avoidance” behaviour
- Otherwise, it will use the output of the “Evasion” behaviour. The target position and target velocity are set to the target position and velocity of the target ship, so it will move away from the target ship (using the target ship’s velocity to predict where the target ship will be in the future)

State Behaviour Combiner

Once the state method has selected the behaviour inputs, a behaviour combiner will iterate through each of the behaviours in turn and use them to create a final output.

Behaviour Combiner	Description
Priority Only	Iterates through the behaviour inputs in order, calculating the outputs of each in turn. The first behaviour that returns a nonzero output is chosen as the final output.
Prioritised Dithering	Iterates through all the behaviour inputs in order. Like “Priority Only”, any behaviour that has a nonzero output is skipped. However, unlike “Priority Only”, each behaviour also only has a given probability of being evaluated, which is determined by weighting. For example, if the weighting is 0.7, the behaviour has a 70% chance of being evaluated (otherwise it will be skipped, no matter whether it has a nonzero output).
Weighted Average	Iterates through all of the behaviour inputs in order, adding outputs from all of the behaviours based on their weighting. The final output is a weighted average of all of the behaviour outputs.

Default AI Behaviours

Behaviour Type	Description	Required Inputs
Idle	Comes to a complete stop.	Weighting.
Seek	Moves directly towards target position.	Target position, weighting.
Flee	Moves directly away from target position.	Target position, weighting.
Pursuit	Moves towards the future position of an object currently at target position moving with a velocity of target velocity.	Target position, target velocity, weighting.
Evasion	Moves away from the future position of an object currently at target position moving with a velocity of target velocity.	Target position, target velocity, weighting.
Seek Arrival	Moves directly towards target position, slowing down when nearing target position to come to a complete stop upon reaching it.	Target position, weighting.
Seek Moving Arrival	Moves directly towards target position, changing speed when nearing the target position to match target velocity upon reaching it.	Target position, target velocity, weighting.
Pursuit Arrival	Moves towards the future position of an object currently at target position moving with a velocity of target velocity, changing speed when nearing the target position to match target velocity upon reaching it.	Target position, target velocity, weighting.
Unblock Cylinder	Moves out of an imaginary cylinder. The cylinder starts at the target position, stretches out infinitely in the direction of target forwards and has a radius of target radius. If the ship is not in the cylinder, returns a zero output.	Target position, target forwards, target radius, weighting.
Unblock Cone	Moves out of an imaginary cone. The cone starts at the target position, stretches out infinitely in the direction of target forwards, and the angle between its central axis and its edges is target FOV angle. If the ship is not in the cone, returns a zero output.	Target position, target forwards, target FOV angle, weighting.
Obstacle Avoidance	Takes preventative action to avoid obstacles. If the ship does need to take preventative action, returns a zero output.	Weighting.
Follow Path	Moves onto and then follows the target path.	Target path, weighting.
Dock	Moves directly towards target position and (when it gets within target radius) attempts to match orientation of target forwards and target up. Target velocity indicates the velocity of the target position (set it to Vector3.zero if it is not moving).	Target position, target forwards, target up, target radius, target velocity, weighting

For an example of writing a custom Behaviour, see Demos\scripts\DemoFlyToLocation.

SSC Radar

This futuristic radar system centrally collects all data from items that send data to it, much like Automatic Dependent Surveillance – Broadcast (ADS-B) does today. At any time, items “in the system” can determine to become invisible to other objects. This impartial system centrally manages all radar communications.

The radar system can be queried via an API in C# code. Results are returned as “blips” which can be used in our built-in UI mini-map, or be used for whatever purpose you deem is useful to your game. Examples include weapon targeting, friend or foe detection, or guidance systems.

IMPORTANT: There should only ever be one radar system in any scene. If you don’t need to use our built-in UI and mini-map, the radar system will be automatically added to the scene at runtime should it be required.

To use the Radar API, see the appropriate section in “Runtime and API” later in this manual.

General Properties

Property	Description
Initialise on Start	If enabled, the GetOrCreateRadar() will be called as soon as Start() runs. This should be disabled if you are instantiating the SSCRadar through code and using the SSCRadar API methods.
Initial Pool Size	The number of items that you expect to be tracked by the radar system at any point in time. If the number exceeds this, the pool will be automatically expanded (although this will have some performance impact).
Range (metres)	The range of the radar from the centre to the edges. Can be overridden at runtime using API methods.
3D Query	Uses 3D distances to determine range when querying the radar data.
Query Sort Order	The order in which query results are returned. Use None where possible as it the fastest and has the lowest impact on performance.

Visual Properties

The built-in UI or visual components of the radar system are entirely optional and internally run radar queries and process results just like you would with the Radar API. You can configure the UI in the scene by adding a radar system to the scene and configuring the visuals in the radar editor.

Property	Description
Screen Locale	Position where radar will be displayed on the screen.
Custom Locale	X,Y coordinates where radar will be displayed on the screen.
Display Width	The radar display width as a proportion of the screen width.
Overlay Colour	Colour of the overlay decals on the radar display
Background Colour	Primary background colour on the radar display
Blip Friend Colour	When the built-in UI is used, this is the colour of any blip that are considered as friendly. Determined by the factionId when available.
Blip Foe Colour	When the built-in UI is used, this is the colour of any blip that are considered as hostile. Determined by the factionId when available.
Blip Neutral Colour	When the built-in UI is used, this is the colour of any blip that are considered as neutral. Determined by the factionId when available. Faction Id = 0

Property	Description
Mini-map UI Image	A reference in the scene to the UI RawImage to be used to display the radar. This is automatically set when a Mini-map is created in the scene.

Movement Properties

When the built-in on-screen visuals (UI) are in use, the radar can be configured to *move* around with an object or ship or remain at a fixed position.

Property	Description
Ship to Follow	The centre of the radar will move with this ship.
GameObject to Follow	The centre of the radar will move with this gameobject.
Centre Position	The centre of the radar.

Surface Turret Module

Turret weapons on ships are very useful, however, sometimes you may wish to position weapons on a planet's surface. The basic steps to setup a surface turret are:

1. Create an empty gameobject (keep scale to 1,1, where possible)
2. Attach the Surface Turret Module script
3. As child objects, add your turret models
4. Your turret should have a pivot transform around which the turret rotates on the Y-axis. Added this transform to the Turret Pivot Y slot under the Weapon Settings
5. Your turret should also have a gun or barrel pivot as a child of the Y-axis pivot transform. In the editor, this should be added to the Turret Pivot X slot.
6. In code, assign a target for the turret. There is a sample script in Demos\Scripts called SampleSurfaceTurretAssignTarget.cs.

NOTE: Do not attempt to place a Surface Turret on a ship, instead configure a turret weapon on the Combat tab in the ShipControllerModule.

For your convenience, we've included an example prefab in Prefabs\Turrets called Turret2.

Surface Turret - General Properties

Property	Description
Initialise on Start	If enabled, the Initialise() will be called as soon as Start() runs. This should be disabled if you are initialising the turret through code and using the SurfaceTurretModule API methods.
Faction Id	The faction or alliance the item belongs to. This can be used to identify if an item is friend or foe. Default (neutral) is 0.
Squadron Id	Although normally representing a squadron of ships, this can be used on a turret to group it with other things in your scene. Default (unset) is -1.
Auto Create Location	Automatically create a Location in the SSCManager when turret is initialised
Is Visible to Radar	Is this turret (Location) visible to radar queries? NOTE: When health of the weapon reaches 0, it will be removed from the radar system.
Radar Blip Size	The relative size of the blip on the radar mini-map.

Property	Description
Effects Object	The particle and / or sound effect prefab that will be instantiated when the turret is destroyed or reaches 0 health.
Destroy on No Health	Should the turret be destroyed (removed from the scene) when its health reaches 0?
Destruct Object	The destruct prefab that breaks into fragments when the turret is destroyed. Only available if "Destroy on No Health" is enabled.

Surface Turret - Weapon Settings

Property	Description
Weapon Type	(Turret) Projectile or Beam weapon. Beam turrets are currently in Technical Preview.
Relative Position	The position of the weapon in local space relative to the pivot point of the whole turret. To visually modify this in the scene view, ensure the (G)izmo is turned on for this weapon, click the (F)ind button and move with the standard Unity Move Tool.
Multiple Fire Positions	If this weapon has multiple cannons or barrels
Fire Position Offsets	The positions of the cannon or barrel relative to the position of the weapon.
Fire Direction	The direction in which the weapon fires projectiles in local space. +ve Z is fire forwards, -ve Z is fire backwards. To visually modify this in the scene view, ensure the (G)izmo is turned on for this weapon, click the (F)ind button and rotate with the standard Unity Rotate Tool.
Beam Prefab	Prefab template of the beam fired by this weapon. Beam prefabs need to have a Beam Module script attached to them.
Projectile Prefab	Prefab template of the projectiles fired by this turret. Projectile prefabs need to have a Projectile Module script attached to them.
Reload Time	The minimum time (in seconds) between consecutive firings of the weapon.
Power-up Time	The minimum time (in seconds) between consecutive firings of beam weapons.
Max Range	The maximum distance (in metres) the beam weapon can fire.
Unlimited Ammo	Can this (projectile) weapon keep firing and never run out of ammunition?
Ammunition	The quantity of projectiles or ammunition available for this weapon if there isn't unlimited ammo.
Charge Amount	The amount of charge the beam weapon has (0 = no charge, 1 = full charge)
Recharge Time	The time (in seconds) it takes the fully discharged beam weapon to reach maximum charge
Starting Health	The initial health value of this surface turret. This is the amount of damage that needs to be done to the turret for it to reach its min performance.
Turret Pivot Y	The transform of the pivot point around which the turret turns or rotates on the local y-axis
Turret Pivot X	The transform on which the barrel(s) or cannon(s) elevate up or down on the local x-axis
Turret Min. Y	The minimum angle on the local y-axis the turret can rotate to
Turret Max. Y	The maximum angle on the local y-axis the turret can rotate to

Property	Description
Turret Min. X	The minimum angle on the local x-axis the turret can elevate to
Turret Max. X	The maximum angle on the local x-axis the turret can elevate to
Turret Move Speed	The rate at which the turret can rotate
Turret Inaccuracy	When inaccuracy is greater than 0, the turret may not aim at the optimum target position. This can improve the chances of the target not being hit by the weapon. It will have little or no effect if the weapon uses a guided projectile.

Surface Turret - Gravitational Properties

Gravitational acceleration and direction can affect how a projectile behaves after it has been fired from the weapon.

Property	Description
Acceleration	The acceleration due to gravity in metres per second squared. Earth gravity is approximately 9.81 m/s per second.
Direction	The direction in which gravity acts on the ship in world space.

Surface Turret – Optional Components

The Surface Turret Module also supports the addition of one or more of the following components:

Component	Description / Benefit
Auto Targeting Module	When attached, this component will automatically assign targets to the Surface Turret's weapon.
Damage Receiver	When attached, this component will automatically transmit damage to the Surface Turret's weapon if a collider (on the same gameobject) is hit by a projectile.

Auto Targeting Module

The Auto Targeting Module is designed to be attached to either a ship (Ship Control Module) or a Surface Turret Module.

For ships, the weapons need to be:

- Turrets with any projectile type
- Turrets with beam weapons
- Fixed Projectile weapons with guided projectiles
- A mixture of both

The module uses the Radar system to automatically allocate targets to the weapons. When allocating targets, it will take into consideration the maximum range of each weapon and optional line-of-sight (LoS) settings.

For weapons on a ship, the “Auto Targeting” option must be enabled for the weapon(s) on that ship you want Auto Targeting to apply to. Not all weapons on a ship need to have “Auto Targeting” enabled.

Some features of the Auto Targeting Module can be configured at runtime. See the “Runtime and API” chapter for more information.

Property	Description
Initialise on Start	If enabled, the Initialise() will be called as soon as Start() runs. This should be disabled if you are initialising the module through code.
Module Mode	The module can operate in two different modes, Ship Control Module or Surface Turret Module mode. The Mode should be set to match the type of module it is attached to.
Ship Display Module	If configured, targeting information can be set to this heads-up display in the scene
Show Targets on HUD	Show the Targets on the heads-up display
Check LoS (New Target)	When acquiring a new target, when enabled, this will verify there is a direct Line of Sight between the weapon and the target.
Update Target Periodically	Whether the target should be reassigned periodically (after a fixed period of time).
Update Target Time	The time to wait (in seconds) before assigning a new target.
Can Lose Target	Whether the current target can be 'lost' through either loss of line of sight or (for a turret) an inability to lock on to the target.
Target Lost Time	How long (in seconds) a target must be invalid for it to be lost (prompting a new target to be assigned).
Require LoS	Whether a target can be 'lost' if line-of-sight is lost.
Require Target Lock	Whether a target can be 'lost' if the turret is unable to lock on to it.

To use Guided Projectiles with Auto Targeting Module, make sure the Projectile Module prefab has “Guide to Target” enabled.

To show weapon targets on the HUD:

1. Add a HUD prefab to the scene (e.g., Prefabs / Visuals / HUD1)
2. On the Ship Display Module configure at least 1 DisplayTarget (leave Show Target unticked)
3. Add the HUD from the scene to the “Ship Display Module” slot on the “Auto Targeting Module”
4. Tick “Show Targets on HUD”

Damage Receiver

This is a simple component that you can add to your own non-ship assets in the scene to receive notification when a projectile or (laser) beam has hit your object. A sample script (demos\scripts\SampleObjectDamage.cs) is included to demonstrate how this can work in your game code.

When this component is added to a SurfaceTurretModule, it automatically transmits health data to the turret weapon.

Destructible Object Module

This module can be used to trigger a DestructModule and/or an EffectsModule when the health of the object reaches 0.

IMPORTANT: If you want a ship, a ship’s damage regions, or a surface turret to take damage, use the features included in the Ship Control Module or the Surface Turret Module. This module’s to be used with regular gameobjects that don’t include those components. Examples could include buildings or destructible props.

Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the Destructible Object Module is enabled through code.
Starting Health	How much 'health' the object has initially.
Use Shielding	Whether this object uses shielding. Up until a point, shielding protects the object from damage
Damage Threshold	Damage below this value will not affect the shield or the object's health while the shield is still active (i.e., until the shield has absorbed damage more than or equal to the shielding Amount value from damage events above the damage threshold).
Shield Amount	How much damage the shield can absorb before it ceases to protect the object from damage.
Recharge Rate	The rate per second that a shield will recharge (default = 0)
Recharge Delay	The delay, in seconds, between when damage occurs to a shield and it begins to recharge.
Destruct Object	The destruct prefab that breaks into fragments when the object is destroyed. See also the Destruct Module chapter.
Destruct Offset	The offset in the forward direction, from the objects gameobject, that the destruct module is instantiated.
Effects Object	The particle and/or sound effect prefab that will be instantiated when the object is destroyed. See also the Effects Module chapter.
Effects Offset	The offset in the forward direction, from the objects gameobject, that the destruction effect is instantiated.
Use Damage Multipliers	Whether damage type multipliers are used when calculating damage from projectiles.
Damage Type A-F	The relative amount of damage a Type A-F projectile will inflict on the object.
Faction Id	The faction or alliance the object belongs to. This can be used to identify if a object is friend or foe. Neutral = 0.
Squadron Id	Although normally representing a squadron of ships, this can be used on a gameobjects to group it with other things in your scene.
Visible to Radar	Is this object visible to the radar system?
Radar Blip Size	The relative size of the blip on the radar mini-map.

Ship Docking and Undocking

Ship docking consists of two main script components, Ship Docking Station and Ship Docking. Player and AI ships can use this feature. It can be configured in the editor and/or at runtime. Often a combination of both is used as can be seen in Demos\TechDemo\Scenes\techdemo2scene1.

Ship Docking Station - Overview

This is a place where multiple ships can depart from or arrive at. It could be a static hangar on the ground, or it could be part of a large capital or mothership. There are four components:

- The docking station gameobject with a Ship Docking Station script attached
- One or more ships with a Ship Docking script attached
- An optional exit Path (created with SSC Manager)

- An optional entry Path (created with SSC Manager)

When a Ship Docking Station is attached to a Ship Control Module (a.k.a. mothership), the entry and exit Path, if defined, moves with the mothership at runtime. For this reason, entry and exit Paths must be only used by one Ship Docking Station at a time. If you have multiple Docking Stations, and want to use entry/exit Paths, you need to create different ones for each Docking Station.

A single Docking Station should not use the same Path for the entry and exit Paths. This is because the start of the entry Path should be near the docking point and the end of the exit Path should be near the docking points. See Docking Point Properties below for more detail. However, multiple docking points on the same Docking Station can use the same entry and exit Paths.

Entry and exit Paths are created in world-space using the SSC Manager in a scene (just like any other Path would be created for Sci-Fi Ship Controller). See “Locations and Paths” under SSC Manager elsewhere in this manual. It is important that entry/exit Path are correctly aligned with the mothership in the scene.

To create the Paths in the Unity editor, add the mothership (or stationary hangar) to the scene, position it where you want it to be when the scene loads, and then create your Paths using the SSC Manager. If you intend to change this starting position or rotation of the mothership when the scene loads at runtime, in code, you will need to tell the Ship Docking Station that the Paths also need to be changed. This can be done with the `InitialiseDockingPointsPaths(Vector3 positionOffset, Quaternion rotationDelta)` method. See Runtime and API section for more information or contact us on either our Unity Forum or Discord channel.

Basic setup instructions:

1. Add a ShipDockingStation script to a gameobject in the scene (this could be a static object like a hangar or a (mother)ship which includes a ShipControlModule component.
2. Configure one or more Ship Docking Points in the ShipDockingStation. These can be manually configured in the Inspector or can be manipulated in the scene view using gizmos and the Unity Move and/or Rotate tools.
3. Add a ShipDocking script to the ships that you wish to dock with the ShipDockingStation. NOTE: A mothership does not need a ShipDocking script unless it too will dock with a bigger ShipDockingStation (this scenario is shown in Tech Demo #2 where the capital ship also has a hangar it departs from).
4. A ship has an Anchor Point. This is the location on the ship that docks with the ShipDockingStation's docking points. Typically, this anchor point would be at the base or bottom of the ship and face downwards so that it could “land” on a Station's docking point. However, many other orientations are possible so that you could say have a ship dock with an airlock of a space station.
5. Ships can start in a scene as Docked or Undocked. Ships can also be assigned to a Docking Point on a docking station.
6. AI Ships can depart from a station using an Exit Path or can enter on an Entry Path. Paths are created in the scene using the SSC Manager and then selected in the ShipDockingStation's docking points.
7. Ships can be assigned or unassigned to a station docking point in game code. They can also change their state from Docked, to Undocking, to Undocked, and/or Docking.
8. Player ships can Dock or Undock using a button configured in the Player Input Module.

Ship Docking Station - General Properties

Property	Description
Initialise on Awake	If enabled, Initialise() will be called as soon as Awake() runs. This should be disabled if you are instantiating the Ship or ShipDockingStation through code and using the Docking API methods.
On Pre Undock	Methods that get called immediately before Undock or UndockDelayed are executed. WARNING: Do NOT call UndockShip from this event.

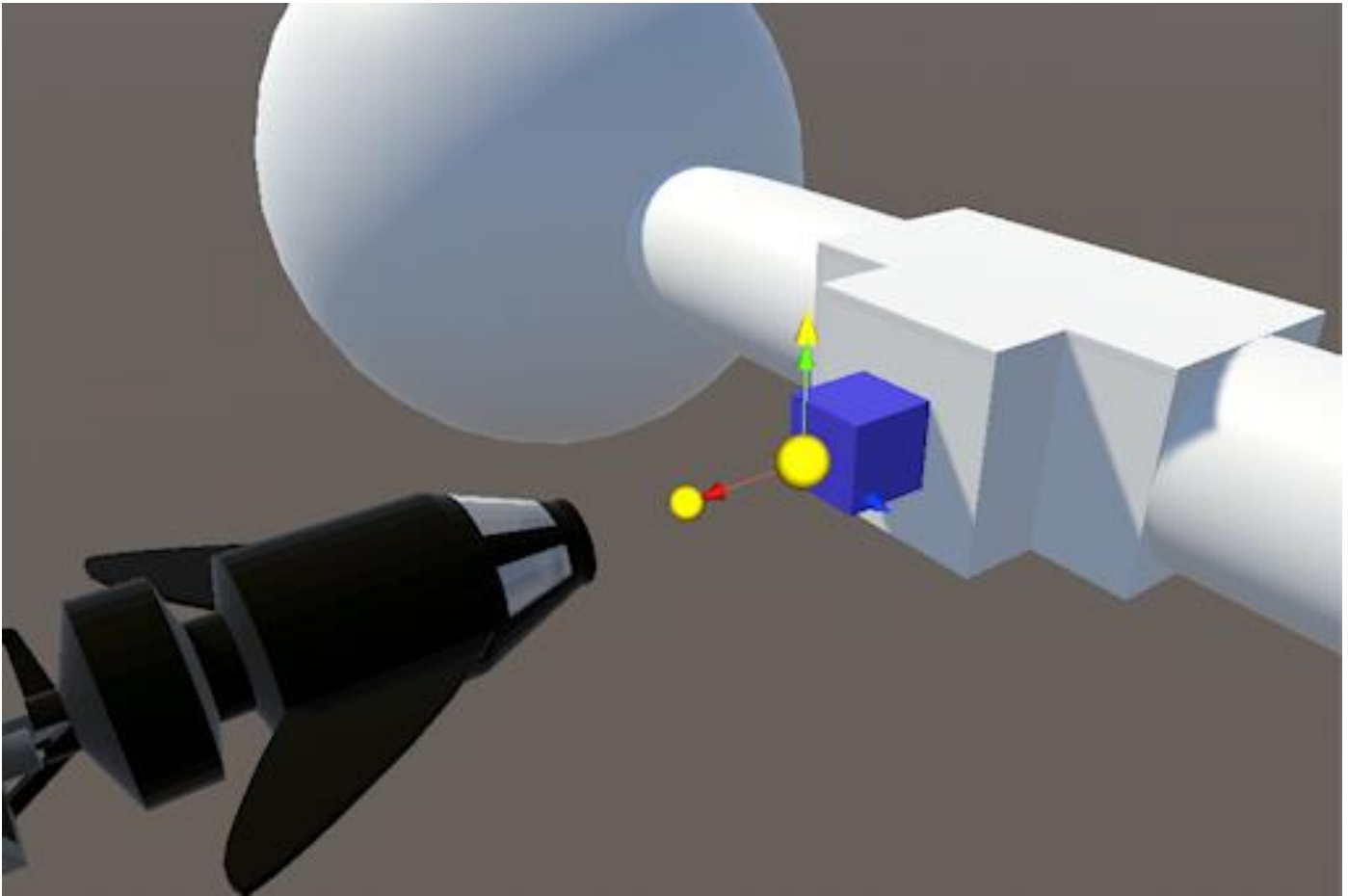
Property	Description
	If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.
On Post Docked	<p>Methods that get called immediately after docking is complete.</p> <p>WARNING: Do NOT call DockShip from this event.</p> <p>If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.</p>

A Ship Docking Station can have 0, 1 or more Docking Points. These are the places where a ship can dock or undock with the docking station. Docking Points show up in the Unity Editor as small clickable gizmos when the “G” button in the inspector is down. When a Docking Point is selected in the scene, it can be moved around like any gameobject. Its relative position can also be updated in the inspector.

Docking points can be exported to a json file or imported using a previously exported file. These json (text) files can be saved in a project or outside of a Unity project. After importing Ship Docking Points, you may still need to configure Docking Entry and Exit Paths, as well as Assigned Ships etc. However, it is a convenient way of copying setting between Ship Docking Stations.

Ship Docking Station – Docking Point Properties

Property	Description
Relative Position	Local position relative to the ShipDockingStation transform position
Relative Rotation	Docking Point rotation in degrees relative to the ShipDockingStation transform rotation
Docking Entry Path	The optional Path (stored as a guidHash) which identifies the entry path a ship can take to dock at this docking point. This is the Path that an AI Ship will automatically take when its docking state is set to “Docking”. The Path must start near the docking point and end where the ship should be considered “undocked”.
Undocking Exit Path	The optional Path (stored as a guidHash) which identifies the exit path a ship can take to depart from this docking point. This is the Path that an AI Ship will automatically take when its docking state is set to “Undocking”. The Path must start when the ship is considered “undocked” and end near the docking point.
Hover Height	This is the optimum height above the docking point in the relative up direction, a ship hovers before arriving or departing.
Assigned Ship	The Ship in the scene that is currently docked, docking, or undocking with the Ship Docking Point. This assigned ship requires a Ship Docking component. It can be a player ship (with a Player Input Module) or an AI Ship (with a Ship AI Input Module).



Ship Docking Component

To dock with a Docking Station, a ship needs to have a Docking Station component attached to it. The ship can either be a player ship (with a Player Input Module) or an AI Ship (with a Ship AI Input Module).

Player ships have a special option in the Player Input Module to allow a user to press a button to dock or undock with a Docking Station Docking Point.

AI Ships can be docked, undocked, docking or undocking. When AI Ships are in a state of “Docking”, they will automatically follow the Docking Entry Path (if there is one defined). When AI Ships are in a state of “Undocking”, they will hover to their desired height then automatically follow the Undocking Exit Path (if there is one defined).

AI Ships are typically assigned to a Docking Point, and put in a specific DockingState, via code.

General Property	Description
Initialise On Awake	<p>If enabled, the Initialise() will be called as soon as Awake() runs. This should be disabled if you are instantiating the ShipDocking through code.</p> <p>If the ship has been assigned to a Ship Docking Station Point in the editor before runtime, you can leave this disabled, as the Ship Docking script will be automatically initialised when the Ship Docking Station is initialised.</p>
Initial Docking State	<p>The ship can start in a Docked or Undocked position. Currently it cannot start docking or undocking. However, in code, you can make an AI Ship start docking or undocking almost immediately a scene loads if need be.</p>

General Property	Description
Landing Distance Precision	How close the ship has to be (in metres) to the docking position before it can become docked.
Landing Angle Precision	How close the ship has to be (in degrees) to the docking rotation before it can become docked.
Hover Distance Precision	How close the ship has to be (in metres) to the hovering position before it is deemed to have reached the hover position.
Hover Angle Precision	How close the ship has to be (in degrees) to the hovering rotation before it is deemed to have reached the hover position.
Detect Collisions (Docked)	Should physics collisions be detected when the state is Docked? Off by default, this could be useful if you want to have characters walk around the ship when it is in the Docked state.
Dock Snap to Pos Axes	The position axes to snap to when a ship gets close to the docking point, and becomes Docked. The snap amount can be affected by the Landing Distance Precision.
Dock Snap to Rot Axes	The rotation axes to snap to when a ship gets close to the docking point, and becomes Docked. The snap amount can be affected by the Landing Angle Precision.
Undocking Delay	When used with ShipDockingStation.UndockShip(..), the number of seconds that the undocking manoeuvre is delayed. This allows you to create cinematic effects or perform other actions, before the Undocking process begins.
Auto Undock Time	When the value is greater than 0, the number of seconds the ship waits while docked, before automatically attempting to start the undocking procedure.
Mothership Undocking	If the Ship Docking Station is attached to big ship, smaller ships may need to immediately have some velocity relative to the moving mother ship.
Undock Vert Velocity	This is additional velocity in an upwards direction relative to the mothership
Undock Fwd Velocity	This is additional velocity in a forward direction relative to the mothership
Catapult Undocking	A catapult can be used to launch a ship from a docking point.
Catapult Thrust (kN)	The amount of force applied by the catapult when undocking in KiloNewtons. For AI and AI-assisted ships, this gets applied when the undocking ship reaches the hover height. For non-AI assisted Player ships, this is applied immediately upon launch.
Catapult Duration	The number of seconds that the force is applied from the catapult to the ship.
Ship Docking Adapter	A ship docking adapter is a location on a ship that allows it to dock with a Ship Docking Point on a Ship Docking Station. In this release, each ship will have 1 adapter.
Relative Position	Local position of the adapter relative to the Ship
Relative Direction	The direction the adapter is facing relative to the Ship. Default is down ($y = -1$). A +ve Z value is forwards, and -ve Z value is backwards. By default, the ship will “land” on top of the Docking Point with the underside of the ship facing the Docking Point on the Ship Docking Station.
Debug Mode	This can be used at runtime in the editor to help determine what is happening in your gameplay. It needs to be enabled while the Unity editor is in play mode.

Event Property	Description
On Dock Start Delay	The time, in seconds, to delay the actioning of any On Post Docking Start methods.
On Post Docking Start	<p>Methods that get called immediately after docking has started. Typically used to perform a non-docking API action like chatter with ground staff, disabling radar, preparing for landing etc.</p> <p>WARNING: Be careful not to call other docking APIs that might create a circular loop.</p>
On (Post) Docking Hover Delay	The time, in seconds, to delay the actioning of any On Post Docking Hover methods.
On Post Docking Hover	<p>Methods that get called immediately after the Hover point is reached when docking. Typically used to perform a non-docking API action like lowering landing gear, disabling radar, disarming weapons etc.</p> <p>WARNING: Be careful not to call other docking APIs that might create a circular loop.</p> <p>If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.</p>
On (Post) Docked Delay	The time, in seconds, to delay the actioning of any On Post Docked methods.
On Post Docked	<p>Methods that get called immediately after the ship has finished docking.</p> <p>If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.</p>
On Undock Start Delay	The time, in seconds, to delay the actioning of any On Post Undocking Start methods.
On Post Undocking Start	<p>Methods that get called immediately after undocking has started. Typically used to perform a non-docking API action like dust or steam particle effects or opening hanger doors.</p> <p>WARNING: Be careful not to call other docking APIs that might create a circular loop.</p>
On (Post) Undocked Delay	The time, in seconds, to delay the actioning of any On Post Undocked methods.
On Post Undocked	<p>Methods that get called immediately after the ship has finished undocking.</p> <p>If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.</p>
On (Post) Undocking Hover Delay	The time, in seconds, to delay the actioning of any On Post Undocking Hover methods.
On Post Undocking Hover	<p>Methods that get called immediately after the Hover point is reached when undocking. Typically used to perform a non-docking API action like raising landing gear, enabling radar, arming weapons etc.</p> <p>WARNING: Be careful not to call other docking APIs that might create a circular loop.</p> <p>If the function you call has four parameters (int, int, int, Vector3), it will appear as a Dynamic Parameters function. This can be useful if you want your method (function) to be automatically passed the ShipDockingStationID, the shipId, and the Docking Station Point Number (1 – n). The Vector3 is for future use.</p>

See also Runtime and API, Ship Docking API Methods.

Player Docking – Non-assisted

Human player ships that use the Player Input Module can dock and undock with a Ship Docking Station. Your player flies their ship really close to the docking point, then presses the button setup in the “Docking Input Button” section of the Player Input Module’s editor. The ship will “snap” to the docking point.

The player ship must also have a Ship Docking component attached to the same gameobject as the Ship Control Module. See “Ship Docking Station – Docking Point Properties” above to configure a docking point for the ship.

Player AI-Assisted Docking

Human player ships that use the Player Input Module can use the Ship AI Input Module to help them dock and undock with a docking station. This is different than using the “Docking Input Button” in the Player Input Module which will immediately set the docking state to either “Docked” or “Undocked”.

The player ship must have the following components:

- Ship Control Module
- Player Input Module
- Ship Docking
- Ship AI Input Module

You can configure the player to either press a single button to toggle docking and undocking or you can configure two buttons, one for docking and the other for undocking.

Single Button Method

1. On the Player Input Module, make sure the “Docking Input Button” is not configured
2. Expand “Custom Player Inputs” and add a new one
3. Configure the button or input axis for your controller or keyboard input
4. Add a Callback Method with the “+” button provided
5. Drag the Player ship gameobject into the Object slot under “Runtime Only”
6. For the “Function” select “Player Input Module” -> ToggleAIDocking()

Two Button Method

1. On the Player Input Module, make sure the “Docking Input Button” is not configured
2. Expand “Custom Player Inputs” and add a new one
3. Configure the docking button or input axis for your controller or keyboard input
4. Add a Callback Method with the “+” button provided
5. Drag the Player ship gameobject into the Object slot under “Runtime Only”
6. For the “Function” select “Player Input Module” -> EnableAIDocking()
7. Add another Custom Player Input
8. Configure the undocking button or input axis for your controller or keyboard input
9. Add a Callback Method with the “+” button provided
10. Drag the Player ship gameobject into the Object slot under “Runtime Only”
11. For the “Function” select “Player Input Module” -> EnableAIUndocking()

NOTE: When configuring buttons for docking and undocking, ensure the button is not configured to be held down. You want the event to only occur once.

Ship Display Module (HUD)

The Ship Display Module can be used as a heads-up display (HUD) for your player ship.

This feature is currently **NOT SUPPORTED with VR** because Unity does not support screen space overlay in VR.

Prefabs included in the Prefabs\visuals folder can be dropped into a scene. If you plan to modify the prefab, we suggest creating a copy or “original” prefab from this one to work with. Otherwise, when you next update Sci-Fi Ship Controller you may overwrite your changes.

When making changes to a HUD prefab, ensure you edit in the scene OR “Open Prefab”. If you attempt to make changes when the prefab is not open or in the scene you will receive many errors and warnings in the console which may leave the HUD prefab in an inconsistent state.

The Ship Display Module is a UnityEngine.UI or canvas-based solution. Currently it does not use any custom shaders and therefore should work wherever the UI canvas and components are supported. If you see any platform-centric issues, please let us know.

The module currently has the following feature areas:

- Auto-hide cursor
- Reticle selection
- Altitude and Speed
- Display Attitude (pitch)
- Flicker the HUD on/off
- Display Heading (compass)
- Display Messages
- Display Targets
- Display Gauges
- API for integration with your own project

Our plan is to add new features over time based on typical user requirements. Where possible, we endeavour to add core features that have widespread appeal. We believe stable low-level features are more important than visual or graphical elements. That’s because we know you will have very particular look and feel requirement which will set you game or project apart from others. Essentially, we want to empower you rather than restrict you.

Don’t feel compelled to use this module. If you want to design a totally different HUD you can still do so and use our extensive API to pull data from ships, radar, weapons, health, damage etc.

WARNING: Do not manually change the size, anchor points or settings of the HUD elements in the scene. This could lead to unpredictable results.

Ship Display Module – Extending

As the module is constructed on top of the standard Unity UI and has its own built-in API, you should be able to extend its functionality without too much fuss.

We’d recommend the following approach:

1. Create a custom MonoBehaviour script in your own namespace
2. Attach this component to the same GameObject as the ShipDisplayModule component
3. In your script get a reference to both the ShipDisplayModule and the Canvas.
4. Call any ShipDisplayModule API methods as required
5. Add UI elements giving them unique names on the same canvas.
6. Update your UI elements as required

Ship Display Module – General Settings

Many of these properties can be adjusted in real-time in the editor at runtime to see the effect they have.

Property	Description
Initialise on Start	If enabled, the Initialise () will be called as soon as Start () runs. This should be disabled if you are instantiating the HUD through code.
Show on Initialise	Show the HUD when it is first Initialised
Show Overlay	Show the overlay image on the HUD.
Auto Hide Cursor	Automatically hide the screen cursor or mouse pointer after it has been stationary for a fixed period of time. Automatically show the cursor if the mouse is moved provided that the Display Reticle is on shown.
Hide Cursor Time	The number of seconds to wait until after the cursor has not moved before hiding it
Main Camera	The main camera used to perform calculations with the heads-up display. If blank will be auto-assigned to the first camera with a MainCamera tag.
Source Ship	The ship in the scene that will supply the data for this HUD
HUD Width	The head-up display's normalised width of the screen. 1.0 is full width, 0.5 is half width. To see the effect of this outside play mode, enable Show HUD Outline and look in the scene view.
HUD Height	The head-up display's normalised height of the screen. 1.0 is full height, 0.5 is half height. To see the effect of this outside play mode, enable Show HUD Outline and look in the scene view.
HUD Offset X	The head-up display's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen. To see the effect of this outside play mode, enable Show HUD Outline and look in the scene view.
HUD Offset Y	The head-up display's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen. To see the effect of this outside play mode, enable Show HUD Outline and look in the scene view.
Primary Colour	Primary colour of the heads-up display. This changes the colour of the overlay image. This is affected by Brightness. Calibrate when Brightness = 1.
Brightness	This is the overall brightness of the HUD relative to its initial state at runtime.
Canvas Sort Order	The sort order of the canvas in the scene. Higher numbers are on top.
Show HUD Outline	Show the HUD as a yellow outline in the scene view [Has no effect in play mode]. Click Refresh if screen has been resized. This can help to gauge the overall size of the HUD without going into play mode.

Ship Display Module – Display Reticle Settings

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Active Reticle	Show or render the active Display Reticle on the HUD. [Has no effect outside play mode]
Active Display Reticle	The currently selected or displayed reticle. This is used to help aim your weapons in front of the ship.
Reticle Offset X	The Display Reticle's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.

Property	Description
Reticle Offset Y	The Display Reticle's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Reticle Colour	The colour of the active Display Reticle
Lock Reticle to Cursor	Should the Display Reticle follow the cursor or mouse position on the screen?
Sprite (for each Reticle)	The sprite (texture) to be displayed in the heads-up display for this reticle. Samples of these are provided in the Textures\HUD folder. They should be 64x64, white with a transparent background, and have a Texture Type of Sprite (2D and UI).

To create a custom reticle you can perform the following tasks:

1. In an image editor like Photoshop, Corel Paintshop Pro or Gimp, create a new image with a transparent background that has dimensions of 64x64 pixels.
2. Draw your reticle in white (RGBA 1,1,1,1) – you can colour the reticle inside the Ship Display Module.
3. Import the image into Unity (we have used PNG but the format should not matter)
4. Change the Texture Type to “Sprite (2D and UI)”
5. Use this sprite in your Display Reticle.

Ship Display Module – Altitude and Speed Settings

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Altitude	Show the Altitude indicator on the HUD. Typically, only used when near the surface of a planet. In space this would be turned off. Like in an aircraft, it does NOT consider the height of the terrain below it.
Show Air Speed	Show the Air Speed indicator on the HUD. Speed is in km/h.
Ground Plane Height	Used to determine altitude when near a planet's surface. On Earth this would typically be sea-level but it can be used to set an artificial zero-height which may be useful when flying over the surface of a planet. This is the height in metres on the world-space y-axis.
Altitude Text Colour	The colour of the Altitude text (number). This are affected by Brightness. Calibrate when Brightness = 1.
Air Speed Text Colour	The colour of the Air Speed text (number). This are affected by Brightness. Calibrate when Brightness = 1.

Ship Display Module – Display Attitude Settings

These settings are used display a scrollable attitude on the HUD. Typically, the attitude will contain a pitch ladder from -90 to 90 degrees with an artificial horizon bar in the middle.

Property	Description
Show Attitude	Show or hide the Attitude as a scrollable image in the UI.
Scrolling Sprite	The sprite (texture) that will scroll up or down. You can use our sprite or create your own. See Textures\HUD\SSCUIPitchLadder1
Mask Sprite	The sprite (texture) that will mask the scrollable attitude sprite. E.g., Textures\HUD\SSCUIFilled

Property	Description
Height	The normalised masked height of the scrollable attitude. 1.0 is full height of the screen, 0.5 is half height.
Offset X	The Display Attitude's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Attitude's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Primary Colour	Primary colour of the scrollable attitude.
Sprite Border Width	The number of pixels between the top of the scroll sprite and the first (90) pitch line. It is assumed that this is the same for between the bottom and the -90 pitch line. This is the actual number of pixels in the Scrolling Sprite image. It gets scaled at runtime in the HUD.

Ship Display Module – Display Fade Settings

These settings are used to fade the display in or out to a specified solid colour.

Property	Description
Fade In On Start	Fade the display in when it first starts.
Fade In Duration	The length of time in seconds, it takes to fully fade in the display.
Fade Out On Start	Fade the display out when it first starts.
Fade Out Duration	The length of time in seconds, it takes to fully fade out the display.
Fade Colour	The colour the display will fade from or to.

Ship Display Module – Display Flicker Settings

These settings are used to determine how a feature of the module flickers on or off. Currently it applies to HUD visibility but may be extended to other features in the future.

Property	Description
Show HUD with Flicker	Whenever the HUD is shown, should it flicker on?
Hide HUD with Flicker	Whenever the HUD is hidden, should it flicker off?
Default Duration	The time, in seconds, the effect takes to reach a steady state
Min Inactive Time	The minimum time, in seconds, that the effect is inactive or off
Max Inactive Time	The maximum time, in seconds, that the effect is inactive or off
Min Active Time	The minimum time, in seconds, that the effect is active or on
Max Active Time	The maximum time, in seconds, that the effect is active or on
Variable Intensity	The intensity of the effect will randomly change
Smoothing	Smooth the flickering effect. Higher values give a smoother effect. [Only applies when Variable Intensity is enabled]
Max Intensity	The maximum intensity of the effect used during the on cycle when Variable Intensity is enabled. This is a multiplier of the starting intensity of the effect. Value must be between 0.01 and 1.0.

Ship Display Module – Display Heading Settings

These settings are used display a scrollable heading ribbon on the HUD.

Property	Description
Show Heading	Show or hide the Heading or direction as a scrollable ribbon in the UI.
Indicator	Show the small heading indicator
Scrolling Sprite	The sprite (texture) that will scroll left or right. You can use our sprite or create your own. See Textures\HUD\SSCUIHeading1
Mask Sprite	The sprite (texture) that will mask the scrollable heading sprite. E.g., Textures\HUD\SSCUIFilled
Indicator Sprite	The small sprite (texture) that will indicate or point to the heading on the HUD. E.g., Textures\HUD\SSCUIIndicator1.
Width	The normalised masked width of the scrollable heading. 1.0 is full width of the screen, 0.5 is half width.
Offset X	The Display Heading's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Heading's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Primary Colour	Primary colour of the scrollable heading.
Indicator Colour	The small indicator colour of the scrollable heading.

Ship Display Module – Display Message Settings

Display Messages are used to present information to the player. They are displayed on the HUD and can also be created, shown, hidden, moved or scrolled at runtime via our extensive API. See “Runtime and API” for more details.

At runtime, messages are stacking on the UI canvas in the order they appear in the Ship Display Module Inspector list. The first item is on placed on the canvas first, and then the second message, and so forth. You can re-order the list by using the small “V” move button.

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Message	Show the message on the HUD. When a message is created, this is off by design. [Has no effect outside play mode]
Message Name	The name or description of the message. This can be used to identify the message.
Message Text	The text to display in the message. It can include RichText markup. e.g. Bold Text
Offset X	The Display Message's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Message's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Display Width	The Display Message's normalised width. 1.0 is full screen width, 0.5 is half width.
Display Height	The Display Message's normalised height. 1.0 is full screen height, 0.5 is half height.
Fade-in Duration	The number of seconds the message takes to become fully visible.

Property	Description
Fade-out Duration	The number of seconds the message take to fade out of view.
Show Background	Show the Display Message background
Text Colour	Colour of the Message text
Text Alignment	The position of the text within the Display Message panel
Is Best Fit	Is the text font size automatically changes within the bounds of Font Min Size and Font Max Size to fill the panel?
Font Min Size	When Is Best Fit is true will use this minimum font size if required
Font Max Size	The size of the font. If isBestFit is true, this will be the maximum font size it can use.
Scroll Direction	The direction (if any) the text should scroll across the screen.
Scroll Speed	Speed or rate at which the text will scroll across the display.
Is Scroll Fullscreen	Scroll full screen regardless of message width and height.

Ship Display Module – Display Target Settings

These are used to show potential enemy targets or friendly ships to the player. Like the “Active Display Reticle”, they use the Reticles from the list that is available in the Display Reticle Settings section of the editor.

Display Targets can be created, shown, hidden or moved at runtime via our extensive API. See “Runtime and API” for more details.

If you are using more than one Display Target, ensure that there are no overlapping factions or squadrons. Each DisplayTarget should have its own unique group of factions and/or squadrons. When you have more than one DisplayTarget configured, ensure the correct faction and/or squadrons are “included” (see Properties below).

Many of these properties can be adjusted in real-time in the editor at runtime.

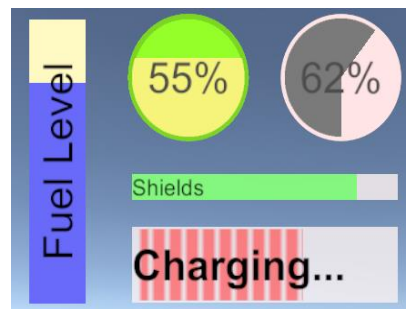
Property	Description
Auto Update Positions	When DisplayTarget slots have an active RadarItem assigned to them, should the reticles be automatically moved on the HUD? Turn this off if you want to move the Display Target reticles yourself in code. It can also be set at runtime.
Show Viewport Outline	Show the rendering limits as a red outline in the scene view [Has no effect in play mode]. Click Refresh if screen has been resized.
Viewport Width	The width of the clipped area in which Targets are visible. 1.0 is full width, 0.5 is half width.
Viewport Height	The height of the clipped area in which Targets are visible. 1.0 is full height, 0.5 is half height.
Viewport Offset X	The X offset from centre of the screen for the viewport
Viewport Offset Y	The Y offset from centre of the screen for the viewport
Targeting Range	The maximum distance in metres that targets can be away from the ship
Per Target Settings	
Show Target	Show the target reticle on the HUD. When a target is created, this is off by design. [Has no effect outside play mode]
Display Reticle	The Display Reticle to use for this Target. To add more Reticles, see the list in the Display Reticle Settings section.

Property	Description
Reticle Sprite	This is what the target will look like on the HUD before colour and brightness is applied.
Reticle Colour	The colour of the active Display Reticle for this Target
Is Targetable	Can this DisplayTarget be assigned to a weapon?
Factions to Include	An array of Faction Ids that the DisplayTarget can show. If the array is empty (0), any item belonging to a faction in the game can be used on this DisplayTarget. This is true except when "Is Targetable" is enabled, in which case only enemy factions can be shown. Using our APIs, you could override this behaviour. If you define it for one DisplayTarget, you must configure it on all DisplayTargets.
Squadrons to Include	An array of Squadron Ids that the DisplayTarget can show. If the array is empty (0), any item belonging to a squadron in the game can be used on this DisplayTarget. This is true except when "Is Targetable" is enabled, in which case only enemy squadrons can be shown. Using our APIs, you could override this behaviour. If you define it for one DisplayTarget, you must configure it on all DisplayTargets.
Max. Number of Targets	The maximum number of these DisplayTargets that can be shown on the HUD at any one time.

Ship Display Module – Display Gauge Settings

These simple measuring bars and gauges can be used to let players know the status of things in your game. Some examples include:

- The health of the player ship
- The amount of charge left in a beam weapon
- Shields level
- Fuel level (assuming you track this)
- The distance to your destination
- Your game score
- Number of times the player can respawn
- The percentage of enemies left to destroy



Gauges are typically constructed by using a different foreground and background sprite (UI texture). There are several examples included in the Textures\HUD folder to get you started. Examples include:

- SSCUICircle1BGnd and SSCUICircle1FGnd
- SSCUICircle2BGnd
- SSCUIFilled (when in doubt, start with this one)
- SSCUIFilledTBBorder1 (has a transparent top and bottom border)
- SSCUIRect1BGnd and SSCUIRect1FGnd
- SSCUIRect2BGnd and SSCUIRect2FGnd
- SSCUIStripeH1Border (has a transparent border)
- SSCUIStripeH1NoBorder
- SSCUIStripeH2Border
- SSCUIStripeH2NoBorder
- SSCUIStripeV1Border
- SSCUIStripeV1NoBorder
- SSCUIStripeV2Border
- SSCUIStripeV2NoBorder

To build custom background and/or foreground sprites, follow the following basic steps:

1. In an image editor like Photoshop, Corel Paintshop Pro or Gimp, create a new image with a transparent background that has dimensions of 64x64 or 256x256 pixels.
2. Draw your sprite details in white (RGBA 1,1,1,1) – you can colour the sprite inside the Ship Display Module. For darker areas use a grey scale. Don't forget to test how they look when the colours are changed in the display module AND the brightness is modified in the General Settings. This is the primary reason why you draw the sprite using white.
3. Import the image into Unity (we have used PNG but the format should not matter). Typically, you will want to place the new sprite in your own folder within the project (not within the SciFiShipController folder).
4. Change the Texture Type to "Sprite (2D and UI)"

There are many API methods that can help you manage and update gauges at runtime.

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Gauge	Show the gauge on the HUD. When a gauge is created, this is off by design. [Has no effect outside play mode].
Gauge Name	The name or description of the gauge. This can be used to identify the gauge.
Gauge Type	The type or style of the gauge. Default: Filled.
Gauge Text	The text to display in the gauge. Does not apply to numeric gauges. It can include RichText markup. e.g., Bold Text
Gauge Label	The label text on a numeric gauge with a label. It can include RichText markup. e.g., Bold Text. For non-numeric gauges, see "Gauge Text".
Gauge Value	The current amount or reading on the gauge. Value must be between 0.0 (empty/min) and 1.0 (full/max).
Offset X	The Display Gauge's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Gauge's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Display Width	The Display Gauge's normalised width. 1.0 is full screen width, 0.5 is half width.
Display Height	The Display Gauge's normalised height. 1.0 is full screen height, 0.5 is half height.
Value Affects Colour	Does the colour of the foreground change, based on the value of the gauge?
Medium Colour Value	When "Value Affects Colour" is true, the value for the foreground medium colour [default: 0.5]
Foreground Colour	Colour of the gauge foreground when the value does not affect the colour.
Foreground Low Colour	Colour of the Gauge foreground when value is 0.0
Foreground Medium Colour	Colour of the Gauge foreground when value is "Medium Colour Value"
Foreground High Colour	Colour of the Gauge foreground when value is 1.0
Foreground Sprite	The sprite (texture) for the foreground of the gauge
Background Colour	Colour of the gauge background.
Background Sprite	The sprite (texture) for the background of the gauge

Property	Description
Fill Method	Determines the method used to fill the gauge foreground sprite when the gaugeValue is modified.
Keep Aspect Ratio	Keep the original aspect ratio of the foreground and background sprites. Useful when creating circular gauges.
Text Colour	Colour of the Gauge text
Text Alignment	The position of the text within the Display Gauge panel
Label Alignment	The position of the label within the Display Gauge panel. This only applies to numeric gauges with a label.
Text Direction	The direction of the text within the Display Gauge panel
Text Style	The style of the text within the Display Gauge panel
Is Best Fit	Is the text font size automatically changes within the bounds of Font Min Size and Font Max Size to fill the panel?
Font Min Size	When Is Best Fit is true will use this minimum font size if required
Font Max Size	The size of the font. If isBestFit is true, this will be the maximum font size it can use.
Is Numeric Percentage	Is the numeric gauge to be displayed as a percentage?
Max Value	When a numeric Gauge Type is used, this is the number to display when Gauge Value is 1.0.
Decimal Places	The number of decimal places to display for numeric gauges.

Debug Mode is used at runtime to show useful information that can help with troubleshooting.

Ship Warp Module

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

Warp Overview

This module can be used for two related purposes. Firstly, to give the appearance of flying rapidly through deep space. This could be like a Star Trek Warp drive effect, flying through some kind of vortex or space-time-tunnel, or create the allusion of a Star Wars style Hyperdrive system.

Secondly, you could use it as part of a cut-scene to fly from space through a planet's atmosphere.

The Warp component should be placed on its own stationary gameobject in the scene. One or more demo prefabs can be located in the SCSM\SciFiShipController\Demos\Prefabs\WarpFX folder.

So how does it work? I'm glad you asked! It works a bit like an old fashion movie set. Think Star Wars in 1977. You have a ship model that remains more or less stationary. You have something that appears to be coming towards you (the two particle systems), the ship seems to be moving around a bit due to some kind of turbulence, and you have background sounds. Oh, and you have a camera which you can control.

And if you want it to do other stuff too, our ShipWarpModule is extendable in code (assuming you are familiar with C# and the Unity Engine).

Warp General Tab

Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the component is enabled through code.
Allow Custom Inputs	Allow the user of Custom Player inputs during warp.
Offset from Ship	The offset, in local space, from warp fx is from the position of the ship.
Max Warp Duration	If greater than zero, the time, in seconds, that warp will automatically disengage.
Night Sky Colour	The colour of the night sky.
Ambient Source	The source of the ambient light. Colour, Gradient or Skybox.
Override Ambient Colour	By default, the ambient sky colour will be set to the Night Sky Colour
Ambient Sky Colour	If overriding the ambient colour, this is the ambient sky colour.

Warp Ship Tab

When configuring maximum shake, pitch and roll settings, less is generally more. Lower values typically will give you more subtle and acceptable results. Next time you are watching your favourite movie or series, notice how much movement actually takes place in similar scenes.

Property	Description
Ship Control Module	The module from the scene used to control the player ship.
Forward Thrust	The amount of proportional thrust to apply to forward thrusters when warp is engaged.
Max Shake Strength	The maximum strength of the ship shake. Smaller numbers are better.
Max Shake Duration	The maximum duration, in seconds, the ship will shake per incident.
Min Shake Interval	The minimum interval, in seconds, between ship shake incidents.
Max Shake Interval	The maximum interval, in seconds, between ship shake incidents.
Max Pitch Down	The maximum angle, in degrees, the ship can pitch down.
Max Pitch Up	The maximum angle, in degrees, the ship can pitch up.
Max Pitch Duration	The maximum time, in seconds, the ship will take to pitch up and down.
Pitch Curve	The curve used to evaluate the amount of pitch over the pitch duration of each pitch incident.
Max Roll Angle	The maximum angle, in degrees, the ship can roll left or right.
Max Roll Duration	The maximum time, in seconds, the ship will take to roll from left to right.
Roll Curve	The curve used to evaluate the amount of roll over the roll duration of each roll incident.

Warp Camera Tab

Property	Description
Ship Camera Module	The module used to control the player ship camera

Property	Description
Apply Settings on Engage	If there are optional camera settings configured, apply the first one when warp is engaged.
Optional Camera Settings	<p>These can be used if you want to cycle or switch between multiple camera settings while warp is engaged. For example, an in-cockpit view and another where the camera is following the ship. See “Ship Camera Settings ScriptableObject” in the “Ship Camera Chapter” for more details.</p> <p>To cycle through the different camera settings, add a Custom Player Input to your player ship’s “Player Input Module”, add a Callback, set the Object to the ShipWarpModule from the scene, and set the Function to ShipWarpModule.CycleCameraSettings().</p>

Warp FX Tab

A good way to start with the particle systems is to look at the demo ShipWarpModule prefabs included.

Property	Description
Inner Particle System	The child particle system used to generate the inner or centre particles for the FX.
Outer Particle System	The child particle system used to generate the outer particles for the FX.
Pause Sound FX	Are the sound effects currently paused? No new sounds will play until it is unpaused.
Sound FX Set *	A set of SoundFX that are randomly selected while warp is engaged.
Max Sound Interval	The maximum interval, in seconds, between sound fx when warp is engaged.
Sound FX Offset	The local space relative offset from the ship used when instantiating Sound FX.
Randomise Sound Volume	Is the volume randomised between 50 percent of the EffectsModule default volume, and the default volume?

* Sound FX Sets are a list of EffectsModules (see the “Effects Module” chapter) in a ScriptableObject that can be created in your game assets folder.

1. Select or create a folder in your game under Assets in your Project panel. Be careful not to use any of the SCSM subfolders.
2. Right-click the folder and select Create, Sci-Fi Ship Controller, Sound FX Set.
3. Give it a meaningful name. e.g., Warp Sound FX
4. Add one or more Effects Module prefabs that only contain Sound FX (i.e., no particle systems).

Warp Events Tab

Events allow you to call you own code or SSC APIs when something happens. For example, you might want to disable other items in your scene before warping. When coming out of a warp, things may need to be turned back on and maybe some other methods in your code executed.

Event	Description
On Change Camera Settings	These methods are called immediately after the camera settings have been changed.
On Pre Disengage Warp	These methods get called immediately before Disengage() is executed.
On Post Disengage Warp	These methods get called immediately after Disengage() is executed.
On Pre Engage Warp	These methods get called immediately before Engage() is executed.

Event	Description
On Post Engage Warp	These methods get called immediately after Engage() is executed.

Scriptable Render Pipelines

Sci-Fi Ship Controller is predominately a script-based package. The core modules can run in any render pipeline. However, there are a number of demo prefabs and assets included in the pack. There is a dedicated folder called “SRP” which contains packages that can be unpackaged to work with Universal and High Definition Render Pipelines.

Please read the SSC_SRP_Readme.txt file in the SRP folder before proceeding.

Proximity Component

This component let you call your game code, SSC API methods, and/or set properties on gameobjects when an object with a collider enters or exits an area of your scene. Essentially, it saves you time from having to write collider trigger code. There is also an option to check the object’s Unity tag.

If no tags are provided, all objects can affect this area. NOTE: All tags MUST exist.

A typical use case is when a ship enters an area and you want to perform some kind of custom task or make something particular happen in your game. It could be triggering an enemy attack when the player ship enters enemy airspace.

To add one to the scene, use the 3D Object -> Sci-Fi Ship Controller menu to create a new gameobject with either a sphere or box trigger collider.

Ship Proximity Component

This component let you call your game code, SSC API methods, and/or set properties on gameobjects when a ship enters or exits an area of your scene.

To add one to the scene, use the 3D Object -> Sci-Fi Ship Controller menu to create a new gameobject with either a sphere or box trigger collider.

You can easily restrict which ships can trigger entry or exit events or callbacks. This component does not restrict which ships can enter or exit, rather it notifies you when that happens.

An example use of this component is in Tech Demo 4 (techdemo4scene2). It is attached to a landing pad to automatically dock (land) a player dropship when it gets within range of the docking point.

If you are trying to track where ships are in your scene, we recommend using SSC Radar which can be configured and queried in code.

Property	Description
Initialise on Start	If enabled, Initialise () will be called as soon as Start () runs. This should be disabled if you want to control when the component is enabled through code.
Initialise Delay	How many seconds to delay initialisation if Initialise On Start is true
No Notify Duration	The number of seconds, after initialisation, that events or callbacks will not be triggered by a ship entering or exiting the area. This can be useful if you do not want a ship within the collider area to immediately trigger event or callback notifications when the component is initialised.
Unity Tags	Array of Unity Tags for ships that affect this collider area. If none are provided, all ships can affect this area. NOTE: All tags MUST exist.

Property	Description
Factions to Include	An optional array of ship Faction Ids to detect when entering the area.
Factions to Exclude	An optional array of ship Faction Ids to ignore when entering the area.
Squadrons to Include	An optional array of ship Squadron Ids to detect when entering the area.
Squadrons to Exclude	An optional array of ship Squadron Ids to ignore when entering the area.
On Enter Methods	Methods that get called when a ship enters the trigger area. If you create a method in your game code that take 4 parameters (int shipId, int factionId, int squadronId, int proximityComponentId), it will appear in the event function list as a dynamic method.
On Exit Methods	Methods that get called when a ship exits the trigger area. If you create a method in your game code that take 4 parameters (int shipId, int factionId, int squadronId, int proximityComponentId), it will appear in the event function list as a dynamic method.

In code, you can also get notified by creating a custom method in your game code, and assigning it to “callbackOnEnter” or “callbackOnExit”. E.g., callbackOnEnter = MyNotificationMethod;

SSC Moving Platform

This component helps you control the movement and rotation of a platform. This can be helpful when you want to create a lift or elevator mechanism to raise, lower, or transport, a spaceship, vehicle, equipment, or even a 3rd party character controller from A to B.

NOTE: This is *not* a full elevator system and has no concept of floors.

Property	Description
Initialise on Start	If enabled, Initialise () will be called as soon as Start () runs. This should be disabled if you want to control when the platform is enabled through code.
Move	Does the platform move?
Relative Positions	Use positions relative to the initial gameobject position, rather than absolute world space positions.
Average Move Speed	Average movement speed of the platform in metres per second
Wait Time	The time the platform waits at each position
Movement Profile	The *profile* of the platform's movement. Use this to make the movement more or less smooth.
Smooth Start Time	The maximum time it takes the platform to come to resume normal speed when smooth start is used with StartPlatform().
Smooth Stop Time	The maximum time it takes the platform to come to a stop when smooth stop is used with StopPlatform().
Rotate	The starting rotation of the platform in degrees
Starting Rotation	Does the platform rotate?
Rotation Axis	The axis of rotation of the platform.
Rotation Speed	The rotational speed of the platform in degrees per second.

Property	Description
Audio Source	The audio source containing the clip to play for the platform. Must be a child of the platform gameobject.
Overall Audio Volume	Overall volume of sound for the platform
In Transit Sound	The sound that is played while the platform is moving
Arrived Start Sound	The sound that is played when the platform arrives at the first position. This does not play when it is first initialised.
Arrived Start Volume	The relative volume the “Arrived Start Sound” audio clip is played
Arrived End Sound	The sound that is played when the platform arrives at the last position
Arrived End Volume	The relative volume the “Arrived End Sound” audio clip is played
On Arrive Start	These events are triggered by a moving platform when it arrives at the start position.
On Arrive End	These events are triggered by a moving platform when it arrives at the end position.
On Depart Start	These events are triggered by a moving platform when it departs from the start position.
On Depart End	These events are triggered by a moving platform when it departs from the end position.
Positions	A list of 3D positions that the platform will move between. By default, they are in world-space. When “Relative Positions” is enabled, they are relative to the initial position and orientation of the platform.

Moving Platforms support an optional (kinematic) Rigidbody which can be attached to the same gameobject.

SSC and Unity Physics (DOTS)

We have started experimenting with the DOTS version of Unity Physics. It may be a long time, if ever, that this is fully supported in this asset. At the time of writing, DOTS code is rapidly changing with every release and Unity may break major functionality or components with each release.

Nevertheless, we are slowly adding interaction between some SSC components and Unity Physics (DOTS).

When Projectiles are enabled for DOTS, they will be able to collide with both regular colliders and Unity Physics (DOTS) colliders. This can be seen in the Asteroid Demo scene when “AsteroidField” has “Use DOTS” enabled, DOTS for projectiles has been setup, and the Unity Physics (DOTS) package is installed.

To install Unity Physics (DOTS) perform the following tasks:

In Unity 2020.3 LTS

1. Open Project Settings, and select Package Manager
2. Under “Advanced Settings” turn on “Enable Preview Packages” and “Show Dependencies”
3. Close Project Settings, and open the Package Manager
4. Change to Packages to “Unity Register”
5. "Add package from git URL..." under the + menu at the top left of the package manager.
6. Enter “com.unity.physics” and click “Add” (Unity Physics 0.6.0-preview.3 or newer should be installed)

If you have any questions or suggestions about Unity Physics for DOTS and SSC integration, please contact us on our Discord channel or our Unity forum.

Common Issues

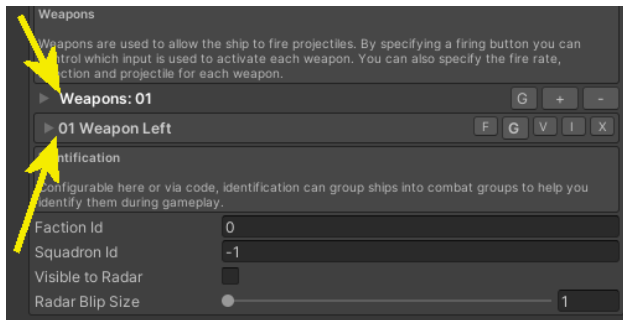
Below is a list of common issues people can encounter that are usually fixed by tweaking the configuration of various components.

Our “Useful Posts” section in our forum might also help:

<https://forum.unity.com/threads/594448/#post-3971293>

Common Issues – General

1. Cannot expand a section of the UI. Click slightly to the right of the arrow near the beginning of the first letter to the right of the arrow. This is a known problem with the newer Unity UI in 2019.1+.



2. All the objects are pink in my scene and/or the terrains in the TechDemo scene are missing. You most likely need to unpack the appropriate Scriptable Render Pipeline package. See the previous chapter in the manual.
3. Setting `Time.timeScale` to 0 produces issues.

Before setting `Time.timeScale` to 0, call:

```
shipCameraModule.MoveCamera();
shipCameraModule.DisableCamera();
shipControlModule.DisableShip(..) // for each ship
sscManager.PauseBeams()
sscManager.PauseDestructs()
sscManager.PauseProjectiles()
sscManager.PauseEffectsObjects()
```

After setting `Time.timeScale` back to 1, call:

```
sscManager.ResumeBeams()
sscManager.ResumeDestructs()
sscManager.ResumeProjectiles()
sscManager.ResumeEffectsObjects()
shipControlModule.EnableShip(..) // for each ship
shipCameraModule1.EnableCamera();
```

See also `PauseGame()` and `UnPauseGame()` in `TechDemo2.cs` and `TechDemo3.cs`.

Common Issues – Auto Targeting Module

1. When I add the Auto Target Module to a ship with a Fixed Projectile Weapon that fires Guided Projectiles, it never targets a Surface Turret. On the “Surface Turret Module” make sure “Auto Create Location” is NOT enabled.
2. I’ve set up a HUD and linked it to the “Ship Display Module” in the “Auto Targeting Module” but my targets don’t appear on the screen. At runtime, click “Debug Mode” on the “Auto Targeting Module” and see if anything is obviously wrong. Check that the ships you want to target are visible to radar (at runtime, click Debug Mode on the targetable ships). Not in play mode, check you have a Display Target configured on the

Ship Display Module (HUD) and that “Is Targetable” is enabled. Make sure the you have at least one matching “Factions to Include” or “Squadrons to include” and that they match your ships you wish to target. Ensure you have a weapon on the player ship that has “Auto Targeting” enabled. This needs to be either a guided projectile if using a “Fixed Projectile” type, or the type needs to be a Projectile or Beam Turret.

3. See also “Common Issues – Weapons”

Common Issues – Player Input

1. Rewired is configured but nothing happens in play mode. In the Player Input Module, when the “Input Mode” is set to “Rewired”, the player needs to be assigned a Rewired player number. This can be done in code, by calling `PlayerInputModule.SetPlayerNumber(..)`, or by setting the Player Number in the Player Input Module editor to a non-zero value, like 1,2,3 or 4.
2. My ship doesn’t move or respond correctly to player input. Although there are several causes, a handy feature in play mode in the Unity editor, is enable “Debug Mode” on the Player Input Module. This can help to determine which input is actually being received and is being sent to the ship.
3. I need to be able to open the cockpit canopy or raise/lower the landing gear when the user presses a button. You can perform custom actions using a Custom Player Input. See the Player Input Module. You can trigger animations using this method by combining it with the `SSCDoorAnimation` script. See “Demo Scripts” in the “Runtime and API” chapter.
4. After upgrading Unity Input System from 1.0.1 to 1.0.2 I get the message “The Unity Input System Player Input component requires an Input Action Asset”. This occurs even though an Input Action Asset is already assigned. Restart Unity to correct the issue.
5. How can I add Custom Player Inputs at runtime? See “Player Input Module (General) API Methods” in the “Runtime and API” chapter. Also take a look at `Demos\Scripts\SampleCustomPlayerInput.cs`.

Common Issues – Ship Behaviour

1. Large ships turn too quickly in Arcade mode. In the Ship Control Module editor, on the “Physics” tab, adjust the Pitch, Roll, and/or Yaw acceleration rates. Small numbers will give the user the impression they are piloting a large, heavy spaceship. Default values around 100 are more suited to fighter-style craft.
2. One or more ships seem to jitter or stutter as they move. Ensure that the Interpolation setting on Rigidbody components for ships in your scene near the player ship have the same setting.
3. After I create a new Ship prefab or replace the model on an existing prefab, my ship jitters or vibrates once it gets moving. Ensure you have a least one Collider on your model. On the Physics tab in Ship Control Module, either turn off Set CoM Manually or click “Reset Centre of Mass”. On the Aero tab, click “Calculate Drag Properties”.
4. In Arcade mode, when turning sharply at speed, the ship pitches and rolls erratically over uneven ground when “Stick to Ground Surface” is enabled. Increase the “Ground Normal History”.
5. Ship won’t move forwards. If Gravity acceleration is non-zero it may just move in the direction of Gravity. Thrusters may to be on but have no effect on motion. On the Physicals tab, ensure “Initialise on Awake” is enabled or it is set within a script.
6. The ship goes too fast. Either decrease the Max Thrust of the forward thrusters and/or on the “Aero” tab, increase the Drag Z Coefficient. Change the Drag Z Coefficient to affect top speed but still allow the ship to accelerate quickly from rest.
7. Rotational Flight Assist doesn’t seem to have any effect, even when it is turned all the way up to 10. If using the legacy Unity Input System, make sure “Gravity” for an Axes is not set to 0. Setting it to a high value, like 1000 can quickly test if that is the issue.
8. Thruster sound flickers on and off. Adjust the Throttle Up and Down times on the Thrusters that have an Effects Object containing sound effects.
9. At speed, my ship veers to one side or trends upward or downward when I attempt to fly in a straight line. This may be because your model is not symmetrical and is being affected by aerodynamic drag. You can check this by temporarily ticking “Disable Drag Moments” or setting the “Medium Density” to 0 on the Aero tab. You can also try adjusting the “Centre of Mass” on the Physics tab.

10. Setting `Time.timeScale` to 0 produces “Input torque is { NaN, NaN, NaN }”. Call `shipControlModule.DisableShip(..)` before setting timescale to 0, and `EnableShip(..)` after setting it back to 1. See also example `PauseGame()` and `UnPauseGame()` code in `TechDemo2.cs` and `TechDemo3.cs`.
11. When my ship is docked, nothing seems to be able to collide with it. On the docked ship, go to the Ship Docking component, and enable “Detect Collisions (Docked)”.
12. In Arcade mode, Gravity Acceleration seems to have little or no effect unless I make it some ridiculously large value. On the Physics tab, decrease “Flight Turn Acceleration”.

Common Issues – Ship AI Behaviour

1. My ship suddenly flips upside down while manoeuvring. If your ship is designed to fly in 3D with any rotation, make sure the “Movement Algorithm” on the Ship AI Input Module is set to “Full 3D Flight”.
2. My ship jerks backwards and forwards while trying to follow a path and may actually go backwards. If the Physics Model is Arcade on the Ship Control Module, if turning off “Use Brake Component” on the Aero tab fixes the issue, use a lower “Brake Strength” when “Use Brake Component” is enabled.
3. When following a path, my AI ship can seem to get stuck or come to a stop when near a particular path point. It is likely that the path tangent (made using the 2 control points of a path point), is facing in the wrong direction and ship suddenly thinks the path is going in the opposite direction. Move one of control points to the other side of the path point (kind of like rotating it 180 degrees around the path point).

Common Issues – Ship Camera Module

1. The camera view suddenly inverts or flips upside down when the ship is climbing at an almost vertical angle. E.g., 90 degrees. If the ship being targeted by the camera can fly straight up or straight down, on the Ship Camera Module, turn off “Orient Upwards”. If the “Target Offset Coords” is set to “Camera Rotation” you could try changing this to “Target Rotation”.
2. `transform.position` assign attempt for 'PlayerCamera' is not valid. Input position is { NaN, NaN, NaN }. The most likely cause is that you set `Time.timeScale = 0f` without calling `shipCameraModule.DisableCamera()`.
3. When “Clip Objects” is enabled and the ship is moving quickly (say 200+ m/s) artifacts of the ship appear in front of the ship (especially in an in-cockpit view). To resolve, either click the “Est.” next to “Minimum Distance” or increase the “Minimum Distance”.
4. When “Lock to Pos” and/or “Lock to Rot” are set, the camera may not be perfectly locked. Check the inspector at runtime. The camera might be using Fixed Update rather than Late Update.

Common Issues – Destruct Module

1. Currently there are no known issues.

Common Issues – Demo Scenes

1. Everything is pink. This occurs when the material shaders in the scene cannot be rendered. The most common cause is that the project was created with a different render pipeline like Universal Render Pipeline (URP) or High-Definition Render Pipeline (HDRP). To apply the correct materials, load the appropriate package from the `SciFiShipController\SRP` folder. See the readme file in this folder for more details.
2. In `TechDemo3`, when `Sticky3D Controller` is also included in the project, I get errors when running the scene. You can either download the “SSC TechDemo3 With Sticky3D” package from the Beta Program, or follow the instructions at the top of the `Techdemo3.cs` script.

Common Issues – Demo Prefabs and Scripts

1. By default, the Celestials (background stars) use the Unity Layer #25. On (rare) occasions, this may conflict with another asset that is difficult to change. To change the layer, open `Demos\Scripts\Celestials.cs` and following the instructions at the top of the file.

Common Issues – Effects

1. Sound on my effects prefab always ignores Volume Rolloff and Max Distance settings. This may be because Spatial Blend is set to 0. Try setting it to 1.

2. Some or all of my particle effects only play once when “Use Pooling” is enabled. This may be because one or more particle systems have a “Stop Action” other than “None”.
3. My thruster produces too few particles when only a small amount of thrust input is received. If at maximum thrust the effect looks correct, in Ship Control Module, on the Thrusters tab, increase the “Minimum Effects Rate” for that thruster.

Common Issues – Radar

1. The UI or mini-map is always shown when the scene first loads. If you want it to be hidden on loading, in the Radar inspector, untick “Initialise on Start” and in your game code call `sscRadar = SSCRadar.GetOrCreateRadar()`. When you are ready to show the UI, call `sscRadar.ShowUI()`.
2. The UI or mini-map does not move or rotate when my player ship moves. In the Inspector of SSC Radar, under “Movement” add your ship to “Ship to Follow”, or call the `sscRadar.FollowShip(..)` API at runtime.
3. On the UI or mini-map, all my friendly and foe blips appear white. The radar compares the blips with faction Id of the ship being followed by the radar mini-map. In the Inspector of SSC Radar, under “Movement” add your ship to “Ship to Follow”, or call the `sscRadar.FollowShip(..)` API at runtime.

Common Issues – Weapons

1. When DOTS is enabled for projectiles, they hit colliders with “Is Trigger” enabled. Either add those objects they hit to the “Ignore Raycast” Unity Layer in the scene, or switch to Pooling.
2. Particle effects (like a Trail) on a projectile prefab don’t appear when DOTS is enabled on the Projectile Module. Currently there is little or no performance benefit for adding Particle Systems (and therefore GameObjects) to DOTS-enabled projectiles. To use particle systems attached to projectiles, switch to Use Pooling. As DOTS matures, we’ll be investigating new solutions in this area.
3. My turret doesn’t fire in the correct direction. If the turret barrel or gun in the editor, does not face towards the front of the ship, you will need to rotate the Fire Direction. In this manual look under Ship Control Module, Combat Tab, Weapons and Fire Direction.
4. I want my non-Turret weapon to fire where the user points on the screen. To do this, convert the screen space coordinates into world-space 3D co-ordinates, then call the following:

```
shipControlModule.shipInstance.SetWeaponFireDirection(weaponIdx, wsTargetPosition);
```
5. My own objects aren’t destroyed when a projectile hits them. Add a Damage Receiver component to your non-ship objects then, in code, decide what to do when the projectile hits the object. A sample script (demos/scripts/SampleObjectDamage.cs) is included to demonstrate how this can work.
6. When a Beam weapon is fired and it hits an object, the Effect stops while the Beam is still hitting the object. To correct this, ensure the Effect Module has a Despawn Time greater than the Max Beam Duration AND the particle effect has “Looping” enabled.
7. My Auto-Fire turrets don’t rotate to aim at the enemy. Check that you have a correctly attached and configured “Auto Targeting Module” on the Surface Turret or Ship. Using “Debug Mode”, at runtime, check that the Auto Targeting Module is initialised and that the targets are in range. Using “Debug Mode” on the ShipControlModule, check that the ship is initialised, the ship is enabled, Movement is enabled, and Health is greater than 0.
8. My Muzzle FX don’t follow the fire point of my weapon. On the Effects Module for the muzzle FX, ensure “Use Pooling” and “Is Reparented” are both enabled.

Common Issues – Moving Platforms

1. My platform jitters when it is close to my ship or third-party character controller. Try adding a (kinematic) Rigidbody to the platform (the SSC Moving Platform will automatically take this into account). If this does not fix the issue, make sure your other moving object has “Interpolate” set on its rigidbody.

Common Issues – Paths

1. When creating a Path in empty space, it can be difficult to add the points where I want them. To solve this, add the first point to the scene by setting the current window to the Scene view and pressing the + key. Then right click in the scene, select “Extend Active Path” from the menu, and drag the new Path Point using the 3D handle provided.
2. I cannot see my path points to edit. On the Path tab, try toggling the (G)izmos icon on/off for the Path.

Common Issues – Ship Docking Station

1. When I manually move the Ship Docking Station in the scene in the Unity Editor, the entry and exit paths do not move with it. Either move the docking station to where it will start before creating the entry and exit paths OR move the paths, one at a time. To move them, select SSC Manager, go to the Path tab, right-click in scene “Select All (Active Path)”, then holding the shift key down and dragging one of the path points in the scene to move the whole path.
2. When undocking, my ship immediately crashes into the mothership. Ensure that you have left, right, up, and down thrusters on your smaller ship. They also need to have enough power (max thrust) to control the smaller ship that is attempting to undock. It can be helpful to first test when the mothership is not moving so that you know everything is setup correctly. It is also possible to give the undocking ship a small boost by modifying the “Undock Vertical / Forward Velocity” settings on the Ship Docking script in the Inspector.
3. When undocking from a moving mother ship, my ship immediately turns to face another direction. This can happen if the docking point relative direction is not in the same direction as the mother ship is facing. One potential solution is to set the undocking ship’s Flight Turn Acceleration to 0 while it is undocking. Then set it back to the original value when the docking state is Undocked. This can be done by setting the `shipDocking.callbackOnStateChange` value to your own method.
4. When undocking, my ship doesn’t reach the hover point. Confirm this by watching the “Debug Mode” for the “Ship Docking” component. If “Is Hover Target” remains true after undocking, then make sure you have thrusters that work in all directions. Forwards, Backwards, Upward, Downward, Left and Right. If the thrusters look correct, adjust the “Hover Distance Precision” and/or the “Hover Angle Precision”.
5. My ship has too much left/right or upward thrust after it has become undocked. Create your own custom method and assign it to the `shipDocking.callbackOnStateChange` delegate. Then set the `maxThrust` value according to the `dockingstate`. `shipInstance.thrusterList[x].maxThrust = xyz Newtons`.
6. My ship gets close to the docking point but never becomes docked. Make sure you have thrusters that work in all directions. Forwards, Backwards, Upward, Downward, Left and Right.
7. My ship gets close to the docking point but keeps moving slightly. That is, it never becomes docked. In play mode in the editor, turn on Debug Mode on the Ship Docking component. Check if the Current State remains “Docking” (it should change to Docked). Check the Ship AI Input Module Debug Mode. If the AI ship also remains in the “Docking” state, look at the “Desired Local Velocity” and “Current Local Velocity”. It is possible one (or more) of your Thrusters does not have enough power to move the ship in the correct direction.
8. My player ship snaps to the docking station point if the player doesn’t get close enough when docking. How can I make this look smoother and more natural? The solution is to use Player AI-Assisted docking. See the “Ship Docking and Undocking” chapter for more details.
9. When my ship is docked, nothing seems to be able to collide with it. On the docked ship, go to the Ship Docking component, and enable “Detect Collisions (Docked)”.

Common Issues – Ship Display Module (HUD)

1. The Altitude or Air Speed indicators are show when I play the scene but their values remain 0. Add a ship from the scene to the “Source Ship” field in the “Altitude and Speed Settings” of the Inspector. This can also be done at runtime via code.
2. The fonts are all wrong – I want something different. Create an original copy of the sample HUD1 prefab. Modify the fonts as required in your new prefab. Make sure you keep the original panel structure and names.

3. When a Display Target is shown on screen, it does not follow the target as the camera position changes. On the Ship Display Module, under “General Settings”, select the correct camera for the HUD.
4. I want a different image for my HUD. On the OverlayPanel, change the “Source Image” to your desired Sprite (UI Texture). Now create an original prefab (if you haven’t already done so) using the HUD in the scene. This will prevent updates to Sci-Fi Ship Controller from overwriting your change.
5. ERROR: Setting the parent of a transform which resides in a Prefab Asset is disabled to prevent data corruption. This occurs if you attempt to edit a prefab from the Project folder without either placing it the scene or without clicking “Open Prefab” in Unity 2019.1 or newer. Contact support if you need help cleaning up your scene.
6. ERROR: InvalidOperationException: Destroying a GameObject inside a Prefab instance is not allowed. This can occur if you try to remove a Message, Target or Gauge from a HUD prefab that has been added to a scene. To correct this, right-click on the HUD gameobject in the scene, select Prefab, and Unpack completely. Remove any unwanted Messages, Targets or Gauges using the HUD Inspector UI. Optionally, to create a new prefab, give the scene HUD gameobject a unique name, and drag it into the Editor Project panel. Click “Original Prefab” when prompted.
7. The Altitude and/or AirSpeed Text do not appear where I want them. If you are using one of the HUD prefabs that comes with SSC, make a duplicate copy of the prefab, rename it, and drag it into your own folder. Then, open your new prefab and, using the normal Unity Editor UI tools, under HUDPanel -> OverlayPanel gameobject, move the “AltitudeText” and/or “AirSpeedText” items to where you want them.
8. When I modify the Gauge Value in the editor or at runtime in code my gauge doesn’t change. Ensure you have added a Foreground Sprite.
9. The HeadingPanel, AttitudePanel or subpanels of these have strange width and heights (e.g., 0 x 0 or say 8498 x 4780). Add the prefab to the scene, on the Ship Display Module, click “Refresh”.

Common Issues – Ship Warp Module

1. When warp is disengaged, I can’t move my ship. This is by design to cater for more varied scenarios. To re-enable ship movement, in the Ship Warp Module Events tab, add a new “On Post Disengage Warp” event, from the scene, add your ship to the “Object” field, and set the “Function” to ShipControlModule.EnableShipMovement().

Common Issues – SSC Celestials

1. The star material is not set. This warning can appear if you have deleted the LBStar.mat material from your project. It is typically located in SCSM\SciFiShipController\Demos\Materials\Environment. If you cannot find it in your project, reimport it using either a full package of SSC or from the Asset Store.
2. The star mesh is not set. This warning can appear if you have deleted the StarLowPolyFBX.fbx model from your project. It is typically located in SCSM\SciFiShipController\Demos\Models\Environment. If you cannot find it in your project, reimport it using either a full package of SSC or from the Asset Store.

Common Issues – VR

1. My hands hit the ship while I’m flying and push it around. See the “Unity XR – Hands and Physics Collisions” section of the “Player Input Module” chapter.
2. How do I use in-game levers or joysticks to control my ship? See the “Unity XR – Levers and Joysticks with Sticky3D” section of the “Player Input Module” chapter.

Runtime and API

SSC is designed to be used in your games. We expect your code to interact with ours. For example, you may wish to do any of the following:

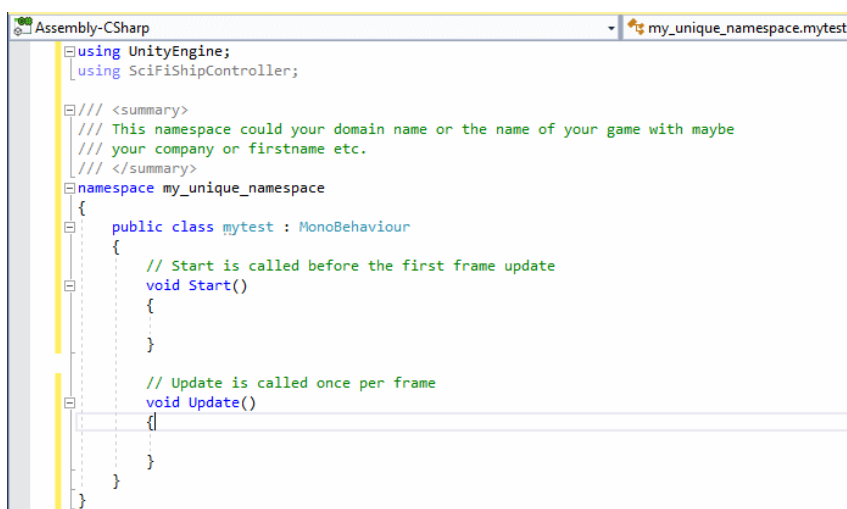
- Spawn Non-Player-Character (NPC) friendly or enemy ships
- Move NPC ships with our built-in Ship Artificial Intelligence (AI) system
- Move NPC ships with your own AI code
- Set targets for turret weapons on ships
- Set targets for surface (ground) turrets
- Update game scoring whenever a ship is destroyed
- Fire weapons automatically from code
- Change the behaviour of a ship as it enters, exits a planet's atmosphere
- Enable or disable "Stick To Ground Surface" in code
- Prevent weapons from firing during certain gameplay
- Assign or unassign a ship from a squadron
- Run queries against the Radar API
- Add or remove Ships and Surface Turrets from radar
- Make your own gameobjects visible to radar
- Assign targets to turret weapons and/or AI ships using the radar data.
- Get an AI ship to follow a path
- Dock or undock Ships on a Ship Docking Station
- Have AI ships attack enemy ships or surface turrets
- Switch the camera module to different ships

If there is anything that you'd like to control at runtime but cannot, please talk to us about that, we're want to provide you the best runtime experience possible.

Runtime General Guidance

The majority of our code is well documented and broken down in to regions marked with #region #endregion tags. These are expandable in Visual Studio.

When integrating Sci-Fi Ship Controller into your game or project, make sure your scripts are in your own namespace so that they don't conflict with other people's code or assets.



```

Assembly-CSharp
using UnityEngine;
using SciFiShipController;

/// <summary>
/// This namespace could your domain name or the name of your game with maybe
/// your company or firstname etc.
/// </summary>
namespace my_unique_namespace
{
    public class mytest : MonoBehaviour
    {
        // Start is called before the first frame update
        void Start()
        {
        }

        // Update is called once per frame
        void Update()
        {
        }
    }
}
  
```

Public Variables and Properties in our scripts are generally available for you to safely access in your own code. Anything marked “[INTERNAL ONLY]”, “private” or “internal” should never be used in your code as these items are subject to change and will most likely either break your game or make it behave in a strange manner.

Some of our scripts have Public API methods. These are used in our demo scripts and can safely be used in your game code. Look for these Public API regions at the bottom of our scripts.

```
// Sci-Fi Ship Controller. Copyright (c) 2018-2020 SCSM Pty Ltd. All rights reserved.
namespace SciFiShipController
{
    [AddComponentMenu("Sci-Fi Ship Controller/Ship Control Module")]
    [HelpURL("http://scsmmedia.com/media/ssc_manual.pdf")]
    [RequireComponent(typeof(Rigidbody))]
    public class ShipControlModule : MonoBehaviour
    {
        Public Variables
        Private Variables
        Public Static Version Properties
        Public Delegates
        Private Initialise Methods
        Update Methods
        Private Methods
        Events
        Public API Methods - Initialisation
        Public API Methods - Reset, Enable, Disable Ship
        Public API Methods - Enable, Disable Ship Movement
        Public API Methods - Radar
        Public API Methods - Ship Input
        Public API Methods - Docking
        Public API Methods - Ship AI
    }
    Public Structures
}
```

Many of our public variables, properties, delegate call-backs, and methods are documented in the sections below in this manual. Everything else is documented in our script files. Feel free to contact us in our Unity forum or on our dedicated Discord channel if you are unsure of how a variable or method should be used.

Changing Variable at Runtime

Many public variables are modifiable at runtime from within your own code. Variables are commented so that (a) you know what they do, and (b) you can see if they require a method to be called after changing at runtime. For example, if you change ship.groundMatchResponsiveness or maxGroundMatchAccelerationFactor at runtime, you also need to call ship.ReinitialiseGroundMatchVariables().

Demo Scripts

We have included a collection of helpful scripts that show how certain features can be used in your games or projects. They are subject to change with version upgrades, so are not meant to be used directly in your projects. Instead, the intention is to help you build games with Sci-Fi Ship Controller by providing coding examples. **Do not** make changes to these scripts, instead create your own based on these.

Most scripts have a description at the top and comments throughout.

Script name	Description
AI Scripts	See also the “Ship AI System” section earlier in this document for other sample Ship AI scripts.
BeaconLight	A script to rotate a light on the y-axis. If an audiosource (and clip) is attached to the same gameobject, it will activate and deactivate at the same time as the light. If including an audiosource, turn off “Play On Awake” for the audiosource. Assumes original rotation y is between -359.999 and +359.999

Script name	Description
Celestials	<p>This sample script, is used to create background stars and planets. Drag the Prefab of the same name into a scene, set the camera and click play. The second camera can be used on a second monitor.</p> <p>NOTE: Currently stars are not rendered in LWRP nor HDRP due to a Unity restriction on multiple cameras.</p> <p>For URP support you need Unity 2019.4+ and URP 7.3.1+.</p>
DemoDockingStation	<p>Demo script that uses your pre-configured Docking Station to undock, then redock AI Ships. You should attach this script to your Ship Docking Station to test that the ships can undock and then re-dock successfully. See the script for more comments.</p> <p>An example use of this script is in the Docking Demo scene.</p>
DemoDockingTransit	<p>Demo script that uses your pre-configured Docking Station for an AI Ship to undock, fly along a path, then dock with another Docking Station. This is designed to work with TechDemo4scene2 but a similar script could be adapted to work with your scenario.</p>
DemoFloatingPoint	<p>Simple demo that acts a bit like a floating-point error manager. It is currently used in the Demos\scenes\Demo Floating Point scene. See also Demos\Scripts\SampleTelePortWorld which moves Paths and AI Ships.</p> <p>NOTE: This is NOT designed to solve all FP error issues.</p>
DemoFlyToLocation	<p>Demo to spawn an AI Ship (or get it from the scene) and head towards the first target location using a custom AIState and custom behaviour. Includes an example of a custom arrival behaviour to slow down a ship as it approaches the location in the scene. The "DemoLocation" is script placed in the scene on another gameobject.</p>
DemoSimpleLOD	<p>A demo script that can be attached to an object to enable or disable the renderer at a given distance from the camera.</p>
DemoSimpleGroupLOD	<p>A demo script that can be attached to a parent gameobject to enable or disable the child renderers at a given distance from the camera.</p>
DemoSSCChangeMaterial	<p>Simple demo script to change 1, 2 or 3 materials of an object, with alternate materials from 3 material arrays. We use it to change lift control panel materials in TechDemo3.</p>
DemoSSCCycleMaterials	<p>Demo script to cycle through two or more materials on an object.</p>
DemoSSCSequenceRenderers	<p>Enable and disable, in a sequence 1, 2 or 3 groups of renderers. In each group, renderers will turn on and off one at a time. An example could be a set of "lights" (using emitting materials) that chase across a landing pad. We use it as a landing pad indicator in TechDemo4.</p>
HideCursor	<p>Sample script hide/show the mouse pointer due to mouse (in)activity. Drop it onto a gameobject in the scene.</p>
SampleAddObjectsToRadar	<p>This simple sample script adds GameObjects to radar. If you hit them with projectiles or (laser) beam from a ship they will be destroyed and removed from radar.</p>
SampleAimWithReticle	<p>This is used with the ShipDisplayModule to aim fixed weapons where the active display reticle is pointing. Attach this script to a gameobject in the scene.</p>

Script name	Description
	NOTE: It is not physically accurate as the fixed weapon on your model will still be pointing in its original direction. e.g., straight ahead. This is only sample to demonstrate how API calls could be used in your own code. You should write your own version of this in your own namespace.
SampleChangeCameraView	This sample script shows how you can use a custom player input action to change the camera view for a ship.
SampleCreatePath	This sample script shows how to create a new Path in the scene at runtime.
SampleDamageRegionHit	Sample code to show how you can get notified when a localised damage region on a ship is hit by a projectile or (laser) beam.
SampleFlyToPosition	Fly an AI ship to a given position in 3D space.
SampleHUDTargets	<p>This sample shows how you can:</p> <ol style="list-style-type: none"> 1. Run radar queries for a player ship to find nearby enemy 2. Use DisplayTargets on a HUD to track enemy on the screen 3. Assign targets to weapons that use Guided Projectiles 4. Get notified when an enemy is hit <p>Setup instructions are included at the top of the script.</p>
SampleInputAIOverride	This simple sample script shows how to override input that gets sent from the ShipAllInputModule to the ShipControlModule of an NPC ship. This script overrides the primary fire mechanism. This script will control when the primary weapons on the AI ship, rather than using something like the AutoTargetingModule.
SampleInputOverride	Attach this script to a player ship to override the Longitudinal axis (forward / backward) player control in the Player Input Module. This sample script demonstrates how one (or more) axis can be overridden while still using the Player Input Module. E.g., control the speed of the player ship.
SampleObjectDamage	Attach a DamageReceiver component to an object in your scene, then use your own code to process the damage received after being hit by a projectile or (laser) beam.
SampleObstacleSpawner	Use by the demo Asteroid Field prefab to create a large number of randomly scaled objects in the scene.
SampleProjectileModule	Demo script used to show how to create your own custom ProjectileModule component.
SampleSendShipInput	<p>Attach a SampleSendShipInput component to a player ship that does NOT use the PlayerInputModule. This is an over-simplified way of sending input to a player ship without using the PlayerInputModule. This is only sample to demonstrate how API calls could be used in your own code. You should write your own version of this in your own namespace.</p> <p>For PC and Console typically, you will use the PlayerInputModule with one of the Input Modes.</p> <p>e.g. Keyboard, Legacy or New Unity Input System, Rewired, Oculus, Vive VR etc</p>
SampleShipSpawner1	Sample ship spawner that leverages the in-build ShipSpawner class. In your project, you'll want to include "using SciFiShipController" namespace and write your own code.

Script name	Description
SampleShowShipHealth	Sample script to show Health of a ship in the UI. This is only sample to demonstrate how API calls could be used in your own code.
SampleShowShipMetrics	Sample script to show how ship metrics like health and beam weapon charge can be added to a Ship Display Module (HUD) at runtime. Setup instructions are included at the top of the script.
SampleShowThrusterMetrics	Sample script to show how thruster metrics can be added to a Ship Display Module (HUD) at runtime. Setup instructions are included at the top of the script.
SampleStickToGroundChange	Sample script used to show how to override the StickToGround options at runtime. This sample controls target ground distance but you could change other options.
SampleTelePortWorld	Sample to create a new Path in the scene at runtime and then teleport the ship and path. This works a bit like a floating-point error manager.
SampleThrusterFXOverride	Sample script to show how to override the thruster particle effects and audio based on the speed of the ship. Attach it to the parent of a ship prefab. This assumes that the particle effects attached to thrusters have Loop enabled.
SampleSurfaceTurretAssignTarget	Sample script to assign a target to a Surface Turret at runtime. This is only a code segment to demonstrate how API calls could be used in your own code. Place it on an empty gameobject in the scene to see how it works.
SampleVariableFlightTurnAccel	Sample component to attach to a ship which dynamically updates the Flight Turn Acceleration of an Arcade ship based on its forward velocity.
SampleWeaponAssignTarget	Sample script to assign targets to turret weapons on a ship at runtime. This is only a code segment to demonstrate how API calls could be used in your own code. Place it on an empty gameobject in the scene to see how it works.
SSCDoorAnimator	<p>See Demos\TechDemo\Scripts folder in the package.</p> <p>Although this is used in our TechDemo scenes, you can probably use this as is in your own games. If you want to make alterations, create your own script so that it won't be overwritten when we do updates.</p> <p>As the name suggests, this can be used to animate one or more sets of doors. It could be used to open/close hangar doors, cargo door(s), bomb-bay doors, raise or lower a cockpit canopy, even raise or lower landing gear!</p> <p>You first need to create your own animation (clips) and one or more Layers in an Animator Controller.</p>  <p>There is some additional information included at the top of the script. If you need assistance, please contact us in the Unity forum or on Discord.</p>

Script name	Description
	Transitions should be conditional based on the value of the isOpen Boolean parameter for that Layer. Each parameter must have a unique name.
SSCDoorControl	<p>This component is used with the SSCDoorAnimator component to open, close or lock a door. Typically, this should be added to the gameobject that contains your door control panel model.</p> <p>See an example in techdemo3scene1. In the scene Hierarchy: Ships, SSCShuttle1 (Arcade), Shuttle SSCDoorControl1.</p>
SSCDoorProximity	<p>This component is used with the SSCDoorAnimator component to trigger when doors are locked or unlocked.</p> <p>Typically, attach to the same object (or child gameobject) of your SSCDoorAnimator component. Then Door Indexes to match the doors in the animator you want to lock or unlock.</p>
SSCObjectRotator	<p>See Demos\TechDemo\Scripts folder in the package. If you want to make alterations, create your own script so that it won't be overwritten when we do updates.</p> <p>This can be used to rotate or pivot an object on 2 axes. An example is the SSCSRadarDish1 included in Demos\TechDemo\Prefabs\Props.</p>

Ship Control Module Methods or Properties

See also Ship Methods or Properties which can be called from shipControlModule.shipInstance.[method].

Property or Method	Description
shipInstance	The class in which the majority of ship data is stored in. Typically use this to access/modify data relating to a ship. You should check IsInitialised before referencing the shipInstance as it is only available on initialised ships.
AddBoost	Add 1 second of forward boost. See also StopBoost(). For more control, see shipInstance.AddBoost(..).
EnableShip (bool resetVelocity)	Enables the ship and make visible. If resetVelocity is set to true, resets the velocity of the ship to zero.
EnableShip (bool updateVisibility, bool resetVelocity)	Enables the ship. If updateVisibility is set to true, makes the ship visible. If resetVelocity is set to true, resets the velocity of the ship to zero. If the ship is enabled for radar, it will become visible to radar.
DisableShip (bool updateVisibility)	Disables the ship. If updateVisibility is set to true, makes the ship invisible. If the ship was enabled for radar, it will become invisible to radar when disabled.
DisableShipMovement()	This is a subset of DisableShip(...). It only applies to Physics, Thruster Effects, User or AI Input to the ship, and Sound FX (audio). See also EnableShipMovement().
DisableRadar()	Similar to EnableRadar() for ships and Locations. If you want a ship or Location to be temporarily invisible to radar, call sscRadar.SetVisibility(..) instead. When using a Localised ShipDamageModel, it will also disable radar on all localised Damage Regions.
DisableRadar (DamageRegion damageRegion)	The ship will no longer send tracking information to the radar system for this damage region. If you want to change the visibility to other radar consumers,

Property or Method	Description
	<p>consider changing the radar item data rather than disabling the radar and (later) calling EnableRadar (damageRegion) again.</p> <p>NOTE: You do not need to call this if calling DisableRadar() for the ship.</p>
EnableShipMovement()	This is a subset of EnableShip(...). It only applies to Physics, Thruster Effects, User or AI Input to the ship, and Sound FX (audio). If a ship is also disabled, this will re-enable the ship. See also DisableShipMovement().
EnableRadar()	<p>shipControlModule.EnableRadar() enables the ship to send tracking information to the radar system. The ship must first be initialised.</p> <p>sscManager.EnableRadar(LocationData locationData) is used to enable a stationary location to be discoverable in the radar system.</p> <p>See also UpdateItem(int itemIndex, SSCRadarPacket sscRadarPacket) in the Radar API Methods section to send radar data from the ship to the radar system.</p>
EnableRadar (DamageRegion damageRegion)	<p>Enable the ship to send tracking information to the radar system for this damage region. The ship must be initialised and radar must already be enabled for the ship. See also EnableRadar().</p> <p>See also UpdateItem(int itemIndex, SSCRadarPacket sscRadarPacket) in the Radar API Methods section to send radar data from the damage region to the radar system.</p>
EnableThrusterEffects (string effectNameContains)	<p>Enable all the thruster effects where the gameobject contains the specified string. Typically used to turn on an effect to improve the quality or look of a game. Call ReinitialiseThrusterEffects() after calling 1 or more of these methods. If you wish to enable or start a thruster, it is more likely you want to use EnableShip() or EnableShipMovement().WARNING: This will generate Garbage so use sparingly. See also StopThrusterEffects ()</p>
DisableThrusterEffects (string effectNameContains)	<p>Disable all the thruster effects where the gameobject contains the specified string. Typically used to turn off an effect to reduce the performance overhead of running it. Call ReinitialiseThrusterEffects() after calling 1 or more of these methods. If you wish to pause or stop a thruster, it is more likely you want to use DisableShip() or DisableShipMovement().WARNING: This will generate Garbage so use sparingly. See also StopThrusterEffects ()</p>
FactionId	Attempt to get or set the faction or alliance the ship belongs to (ship must be initialised). This can be used to identify if a ship is friend or foe. Neutral = 0.
GetShipAllInputModule (bool forceCheck = false)	Retrieves a reference to the ShipAllInputModule script if one was attached at the time this module was initialised. Ignore cached value and call TryGetComponent when forceCheck is true.
GetShipAllInputModule (out ShipAllInputModule shipAllInputModule, bool forceCheck = false)	Retrieves a reference to the ShipAllInputModule script if one was attached at the time this module was initialised. Ignore cached value and call TryGetComponent when forceCheck is true.
GetShipId	Session-only transform InstanceID. This is a fast way of seeing if two references point to the same ship.
GravitationalAcceleration	Attempt to get or set gravitational acceleration (must be initialised)
GravityDirection	Attempt to get or set the direction gravity is acting the ship (must be initialised).
IsInitialised	[READONLY] Has the ship been initialised?

Property or Method	Description
IsRespawning	[READONLY] Is the ship currently being respawned? The ship will be disabled during the respawn time.
IsRespawningPaused	[READONLY] Has respawning been paused? If so, this ship cannot respawn until ResumeRespawning() has been called. See also PauseRespawning().
IsVisibleToRadar	[READONLY] Is this ship visible to the radar?
ResetShip()	A fast way of re-initialising a ship without changing its position or core settings. Typically called when re-initialising a scene or bringing a ship out of hibernation. If you want to temporarily stop a ship from moving, like when a user brings up a menu, call DisableShip() and EnableShip() instead.
NumberOfWeapons	[READONLY] The number of weapons on this ship. Will always return 0 if ship has not been initialised. See also ship.NumberOfWeapons.
NumberOfRespawns	[READONLY] The number of times the ship has been respawned. This is incremented when the ship respawns, not when it is destroyed.
SetThrusterMaxVolume (int thrusterNumber, float newMaxVolume)	Set the maximum volume for a given thruster. Numbers begin at 1. Values should be between 0.0 and 1.0.
ShipIsEnabled()	Returns whether the ship is currently enabled.
ShipIsDocked()	Is the ShipDocking component attached to this ship, and if so, is the ship's state 'Docked'? NOTE: This does not mean it must be docked with a ShipDockingStation. For that, you would need to check the ShipDockingStation's docking points.
ShipIsNotDocked()	Is the ShipDocking component attached to this ship, and if so, is the ship's state 'Not Docked'? NOTE: This does not mean it must be undocked with a ShipDockingStation. For that, you would need to check the ShipDockingStation's docking points.
PauseRespawning()	If the ship is currently respawning, this will stop the countdown timer, preventing the ship from respawning until ResumeRespawning() is called. NOTE: Has no effect if not already respawning.
ResumeRespawning()	If respawning is currently paused, the respawning timer will now continue until the ship is respawned. NOTE: Has no effect if respawningMode is DontRespawn
ReinitialiseThrusterEffects()	Reinitialises the thruster effects for a ship. Call this after modifying thruster effects objects for this ship. WARNING: This will generate Garbage so use sparingly.
ReinitialiseShipBeams ()	Reinitialises variables required for ship beam weapons and effects used by those beams. Call this after modifying any beams or beam effect data for this ship.
ReinitialiseShipDestruct Objects ()	Reinitialises variables required for destruct objects of the ship. Call after modifying any destruct data for this ship.
ReinitialiseShipProjectiles AndEffects()	Reinitialises variables required for projectiles and effects of the ship. Call this after modifying any projectile or effect data for this ship.
ShipRigidbody	[READONLY] The rigidbody of the ship.
ShutdownThrusterSystems (bool isInstantShutdown = false)	Begin to shut down the thrusters. Optionally override the shutdown duration.
SquadronId	Attempt to get or set the (unique) squadron this ship is a member of (ship must be initialised). Do not place friendly and enemy ships in the same squadron.

Property or Method	Description
StartupThrusterSystems (bool isInstantStartup = false)	Begin to bring the thrusters online. Optionally, override the start-up duration. As soon as the systems begin to start up, IsThrusterSystemsStarted will be true.
StopBoost()	Immediately stop any boost that has been applied with AddBoost(..).
TelePort (Vector3 delta, bool resetVelocity)	Teleport the ship to a new location by moving by an amount in the x, y and z directions. This could be useful if changing the origin or centre of your world to compensate for float-point error. NOTE: This does not alter the current Respawn position.
TelePort (Vector3 newPosition, Quaternion newRotation, bool resetVelocity, bool updateColliders = true)	Teleport the ship to a new location with a new rotation. NOTE: This does not alter the current Respawn position. By default, force all attached colliders to update their position relative to the rigidbody in this frame. If false, they will update in the next Physics Update loop which is faster but may cause issues if collider placement is critical.

Ship Control Module API Call Backs

Custom runtime methods should be a lightweight to avoid performance issues. These single-cast delegates can have a single instance of a call-back method. This is useful when you want to take some (custom) action when something occurs, like when a ship is destroyed or is hit or damaged by a projectile.

Property or Method	Description
CallbackOnCollision callbackOnCollision	The name of the custom method that is called immediately after a collision. Your method must take 2 parameters (ShipControlModule and Collision). This should be a lightweight method to avoid performance issues.
CallbackOnDamage callbackOnDamage	The name of the custom method that is called immediately after damage has changed. Your method must take 1 float parameter (mainDamageRegionHealth). It could be used to update a HUD or take some other action. ship.callbackOnDamage = MyOnDamageMethod;
CallbackOnDestroy callbackOnDestroy	The name of the custom method that is called immediately before the ship is destroyed when it reaches a health of 0. Your method must take 1 parameter of class Ship. It could be used to update a score or remove a ship from a squadron. Create your own method in your game code which takes a parameter of the ship class. E.g. Public void MyPreDestroyMethod(Ship ship) { If (ship.someshipvariable == 'something') { DoSomething(); } } .. shipControlModule.callbackOnDestroy = MyPreDestroyMethod;

Property or Method	Description
CallbackOnHit callbackOnHit	<p>The name of the custom method that is called immediately after the ship is hit by a projectile or beam. Your method must take 1 parameter of type CallbackOnShipHitParameters. This should be a lightweight method to avoid performance issues. It could be used to take evasive action while being pursued by an enemy ship. It could also be used to detect friendly fire.</p> <p>shipControlModule.callbackOnHit = MyOnHitMethod;</p>
CallbackOnRespawn callbackOnRespawn	<p>The name of the custom method that is called immediately after the ship is respawned. Your method must take 2 parameters: ShipControlModule and ShipAllInputModule. The first is never null but there may be no AI module attached to this ship. This should be a lightweight method to avoid performance issues.</p>
CallbackOnThrusterSystemsStatusChanged callbackOnThrusterSystemsChanged	<p>The name of the custom method that is called immediately after the thruster systems status has changed. Your method must take 2 parameters: ShipControlModule and int.</p> <p>Status 0 = Starting, 1 = Started, 2 = Shutting down, 3 = Shutdown.</p> <p>This should be a lightweight method to avoid performance issues and should not keep a reference to shipControlModule.</p>
CallbackOnStuck callbackOnStuck	<p>The name of the custom method that is called immediately after a ship is detected as stuck. To avoid performance issues, action should be taken otherwise your method may be called each subsequent frame. See also ship.stuckTime and ship.stuckSpeedThreshold.</p>

Ship (General) Methods or Properties

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
AddBoost (Vector3 forceDirection, float forceAmount, float duration)	Add temporary boost to the ship in a normalised local-space forceDirection which a force of forceAmount Newtons for a period of duration seconds. IMPORTANT: forceDirection must be normalised, otherwise you will get odd results. See also StopBoost().
IsGrounded	[READONLY] Whether the ship is currently sticking to a ground surface.
LocalAngularVelocity	[READONLY] The local angular velocity of the ship as a vector. Derived from the angular velocity of the rigidbody.
LocalVelocity	[READONLY] The local velocity of the ship as a vector. Derived from the velocity of the rigidbody.
PitchAngle	[READONLY] The pitch angle, in degrees, above or below the artificial horizon
ReinitialisePitchRollMatchVariables ()	Re-initialises variables related to pitch and roll match.
ReinitialiseShipPhysicsModel()	Re-initialises all variables needed when changing the ship physics model. Call after modifying shipPhysicsModel.
ReinitialiseGroundMatchVariables ()	Re-initialises variables related to ground distance match calculations. Call after modifying useGroundMatchSmoothing, groundMatchResponsiveness,

Property or Method	Description
	groundMatchDamping, maxGroundMatchAccelerationFactor, centralMaxGroundMatchAccelerationFactor or groundNormalHistoryLength.
ReinitialiseInputControlVariables ()	Re-initialises variables related to Input Control for 2.5D flight. Call this after changing inputControlAxis at runtime.
ReinitialiseThrusterVariables ()	Re-initialises variables related to thrusters. Call after modifying thrusterList.
ReinitialiseWingVariables ()	Re-initialises variables related to wings. Call after modifying wingList.
ReinitialiseInputVariables ()	Re-initialises variables related to ship inputs. Call after modifying thrusterList, controlSurfaceList and/or forceUse / primaryMomentUse / secondaryMomentUse of a thruster or the type of a control surface.
RigidbodyPosition	[READONLY] The rigidbody position of the ship as a vector. Derived from the position of the rigidbody. This is where the physics engine says the ship is.
RigidbodyRotation	[READONLY] The rigidbody rotation of the ship as a vector. Derived from the rotation of the rigidbody. This is where the physics engine says the ship is rotated.
RigidbodyInverseRotation	[READONLY] The rigidbody inverse rotation of the ship as a vector. Derived from the rotation of the rigidbody. This is the inverse of where the physics engine says the ship is rotated.
RigidbodyForward	[READONLY] The rigidbody forward direction of the ship as a vector. Derived from the rotation of the rigidbody. This is the direction the physics engine says the ship is facing.
RigidbodyRight	[READONLY] The rigidbody right direction of the ship as a vector. Derived from the rotation of the rigidbody. This is the direction the physics engine says the ship's right direction is facing.
RigidbodyUp	[READONLY] The rigidbody up direction of the ship as a vector. Derived from the rotation of the rigidbody. This is the direction the physics engine says the ship's up direction is facing.
TransformPosition	[READONLY] The position of the ship as a vector. Derived from the position of the transform. You should use RigidbodyPosition instead if you are using the data for physics calculations.
TransformForward	[READONLY] The forward direction of the ship in world space as a vector. Derived from the forward direction of the transform. You should use RigidbodyForward instead if you are using the data for physics calculations.
TransformRight	[READONLY] The right direction of the ship in world space as a vector. Derived from the right direction of the transform. You should use RigidbodyRight instead if you are using the data for physics calculations.
TransformUp	[READONLY] The up direction of the ship in world space as a vector. Derived from the up direction of the transform. You should use RigidbodyUp instead if you are using the data for physics calculations.
TransformRotation	[READONLY] The rotation of the ship in world space as a quaternion. Derived from the rotation of the transform. You should use RigidbodyRotation instead if you are using the data for physics calculations.
TransformInverseRotation	[READONLY] The inverse rotation of the ship in world space as a quaternion. Derived from the rotation of the transform. You should use

Property or Method	Description
	RigidbodyInverseRotation instead if you are using the data for physics calculations.
StopBoost()	Immediately stop any boost that has been applied with AddBoost(..).
WorldAngularVelocity	[READONLY] The world angular velocity of the ship as a vector. Derived from the angular velocity of the rigidbody.
WorldTargetPlaneNormal	[READONLY] The current normal of the target plane in world space. This is the upwards direction the ship will attempt to orient itself to if limit pitch and roll is enabled.
WorldVelocity	[READONLY] The world velocity of the ship as a vector. Derived from the velocity of the rigidbody.

Ship (Damage) Methods or Properties

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
AddHealth (float healthAmount, bool isAffectShield)	Typically used when ShipDamageModel is Simple or Progressive, add health to the ship. If isAffectShield is true, and the health reaches the maximum configured, excess health will be applied to the shield for the specified DamageRegion. For Progressive Damage, health is also added to components that have Use Progressive Damage enabled.
AddHealth (DamageRegion damageRegion, float healthAmount, bool isAffectShield)	Add health to a specific DamageRegion. If isAffectShield is true, and the health reaches the maximum configured, excess health will be applied to the shield for the specified DamageRegion. For Progressive Damage, health is also added to components that have Use Progressive Damage enabled. NOTE: -ve values are ignored. To incur damage, use the ApplyNormalDamage or ApplyCollisionDamage API methods.
ApplyCollisionDamage (float damageAmount, Vector3 damagePosition)	Applies a specified amount of damage to the ship at a specified position. The damage is registered as "collision" damage, meaning that if it destroys the ship the ship will be respawned at a previously recorded respawn position.
ApplyCollisionDamage (Collision collisionInfo)	Applies damage to the ship due to a collision. The damage is registered as "collision" damage, meaning that if it destroys the ship the ship will be respawned at a previously recorded respawn position.
ApplyNormalDamage (float damageAmount, ProjectileModule.DamageType damageType, Vector3 damagePosition)	Applies a specified amount of damage to the ship at a specified position.
AttachCollider (Collider colliderToAttach)	When attaching an object to the ship, call this method for each non-trigger collider. It is automatically called when using VR hands to avoid them colliding with the ship and causing damage.
AttachColliders (Collider[] collidersToAttach)	When attaching an object to the ship, call this method with an array of non-trigger collider. It is automatically called when using VR hands to avoid them colliding with the ship and causing damage. See also AttachCollider(..) and DetachCollider(..).

Property or Method	Description
DetachCollider (int colliderID)	When detaching or removing an object from the ship, call this method for each non-trigger collider. This is only required if it was first registered with AttachCollider(..). USAGE: DetachCollider (collider.GetInstanceID())
DetachColliders (Collider[] collidersToDetach)	When detaching or removing an object from the ship, call this method with an array of non-trigger colliders. This is only required if they were first attached.
GetDamageRegion (int guidHash)	Get the localised damage region with guidHash in the list of regions. The ship must be initialised.
GetDamageRegionByIndex (int index)	Get the localised damage region with the zero-based index in the list of regions. The ship must be initialised.
GetDamageRegionIndexByName (string damageRegionName)	Get the zero-based index of a local damage region in the ship given the damage region name. Returns -1 if not found or the Damage Model is not Localised. Use this sparingly as it will incur garbage. Always declare the parameter as a static readonly variable. Usage: private static readonly string EngineDamageRegionName = "Engines"; .. int drIdx = GetDamageRegionIndexByName(EngineDamageRegionName); DamageRegion damageRegion = GetDamageRegionByIndex(drIdx);
GetDamageRegionWSPosition (int guidHash)	Get the world space central position of the localised damage region with the given guidHash. If the ship is not using the Localised damage model or a damage region with guidHash cannot be found, this returns the world space position of the ship. The ship must be initialised.
GetDamageRegionWSPosition (DamageRegion damageRegion)	Get the world space central position of the damage region. The ship must be initialised.
HasActiveShield (Vector3 worldSpacePoint)	Is this world space point on the ship, currently shielded? If the main damage region is invincible, it will always return true. If the Ship Damage Model is NOT Localised, the worldSpacePoint is ignored.
MakeShipInvincible()	Make the whole ship invincible to damage. For individual damageRegions change the isInvisible value on the localised region.
MakeShipVincible()	Make the whole ship vincible to damage. When hit, the ship or shields will take damage. For individual damageRegions change the isInvisible value on the localised region.
HealthNormalised	This property returns the normalised (0.0 – 1.0) value of the overall health of the ship. To get the actual health values, see the damageRegions of the ship. This is useful if you want to show the health on the UI for example in a HUD gauge.
IsPointInDamageRegion (DamageRegion damageRegion, Vector3 worldSpacePoint)	Check if a world-space point is within the volume area of a damage region.
LastDamageEventIndex ()	Returns the index of the last damage event. When this value changes, a damage event has occurred.

Property or Method	Description
ReinitialiseDamageRegionVariables (bool refreshAll = false)	Re-initialises variables related to damage regions. Call after modifying mainDamageRegion or localDamageRegionList (or modifying shipDamageModel). NOTE: Not required if just changing the Health or ShieldHealth
RequiredDamageRumbleAmount ()	Returns the amount of rumble required due to the last damage event.
ResetHealth()	Resets health data for the ship. Used when initialising and respawning the ship.
damageRegion.ShieldNormalised	This property returns the normalised (0.0 – 1.0) value of shield for the damage region. If damageRegion.useShielding is false, it will always return 0. For the ship's main shield, use: shipControlModule.shipInstance.mainDamageRegion.ShieldNormalised

Ship (Respawning) Methods or Properties

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
Destroyed ()	Returns true if the ship has been destroyed.
GetRespawnPosition ()	Returns the position for the ship to respawn at.
GetRespawnRotation()	Returns the rotation for the ship to respawn with.
ReinitialiseRespawnVariables ()	Re-initialises respawn variables using the current position and rotation of the ship. Also needs to be called after changing respawningMode to RespawnningMode.RespawnAtLastPosition.

Ship (Thruster) Methods or Properties

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
GetFuelLevel ()	Get the (central) fuel level of the ship. Fuel Level Range 0.0 (empty) to 100.0 (full).
GetFuelLevel (int thrusterNumber)	Get the fuel level of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1. Fuel Level Range 0.0 (empty) to 100.0 (full).
GetHeatLevel (int thrusterNumber)	Get the heat level of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1. Heat Level Range 0.0 (min) to 100.0 (max).
GetMaxThrust (int thrusterNumber)	Get the Maximum Thrust of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1. Values are returned in kilo Newtons.
SetMaxThrustNewtons (int thrusterNumber, float newMaxThrustNewtons)	Use this if you want fine-grained control over max thruster force, otherwise use SetMaxThrust. Set the Maximum Thrust of the Thruster based on the order it appears in the Thruster tab in the ShipControlModule. Numbers begin at 1. Values are in Newtons. This could be useful if you have a very light ship, say a few kilograms, and you need more control.

Property or Method	Description
GetOverheatingThreshold (int thrusterNumber)	Get the overheating threshold of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1.
GetThrusterAudioVolume (int thrusterNumber)	Get the current audio volume for a given Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1.
GetThrusterEffectsObject (int thrusterNumber)	Get the parent gameobject for a given thruster effects. Numbers begin at 1. This is the Effects Object that appears in the inspector of a thruster (if any).
IsThrusterOverheating (int thrusterNumber)	Is the thruster heat level at or above the overheating threshold? Thruster Number is based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1.
RepairThruster (int thrusterNumber)	Repair the health of a thruster. Will also set the heat level to 0. Can be useful if a thruster has burnt out after being over heated. Thruster Number is based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1.
SetFuelLevel (float newFuelLevel)	Set the central fuel level for the whole ship. If useCentralFuel is false, use SetFuelLevel (thrusterNumber, new FuelLevel).
SetFuelLevel (int thrusterNumber, float newFuelLevel)	Set the fuel level of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1. Fuel Level Range 0.0 (empty) to 100.0 (full).
SetHeatLevel (int thrusterNumber, float newHeatLevel)	Set the heat level of the Thruster based on the order it appears in the Thrusters tab in the ShipControlModule. Numbers begin at 1. Heat Level Range 0.0 (min) to 100.0 (max).
SetMaxThrust (int thrusterNumber, int newMaxThrustkN)	Set the Maximum Thrust of the Thruster based on the order it appears in the Thruster tab in the ShipControlModule. Numbers begin at 1. Values should be in kilo Newtons.
SetThrottleAllThrusters (float newThrottleValue)	Set all thrusters to the same throttle value. Values are clamped to between 0.0 and 1.0.
SetThrusterThrottle (int thrusterNumber, float newThrottleValue)	Set the throttle for a given thruster. Numbers begin at 1. Values should be between 0.0 and 1.0.
ShutdownThrusterSystems (bool isInstantShutdown = false)	Use shipControlModule.ShutDownThrusterSystems(..).
StartupThrusterSystems (bool isInstantStartup = false)	Use shipControlModule.StartupThrusterSystems(..)

Ship (Weapon) Methods or Properties

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
AddWeapon (Weapon weaponToAdd)	Add a pre-configured weapon to the current list of weapons and initialise it for use.

Property or Method	Description
DeactivateBeams (SSCManager sscManager)	Deactivate all beam weapons that are currently firing
GetWeaponHeatLevel (int weaponIdx)	Get the heat level of the Weapon with the zero-based index in list of weapons. Heat Level Range 0.0 (min) to 100.0 (max).
GetWeaponIndexByName (string weaponName)	Get the zero-based index of a weapon on the ship given the weapon name. Returns -1 if not found. Use this sparingly as it will incur garbage. Always declare the parameter as a static readonly variable. private static readonly string WPNturret1Name = "Turret 1"; .. int wpldx = GetWeaponIndexByName(WPNturret1Name);
GetWeaponByIndex (int weaponIdx)	Get the weapon on the ship from a zero-based index in the list of weapons. This will be 1 less than the number shown next to the weapon on the Combat tab. It validates the weaponIdx so, if possible, don't call this every frame.
NumberOfWeapons	[READ ONLY] The number of weapons on this ship.
ClearWeaponTarget (int weaponIdx)	Clears all targeting information for the weapon. This should be called if you do not know if the target is a ship or a gameobject. WARNING: For the sake of performance, does not validate weaponIdx.
HasWeaponTarget (int weaponIdx)	Has the weapon with the zero-based index in list of weapons got a target assigned to it? The target could be a GameObject, Ship, or Ship Damage Region. WARNING: For the sake of performance, does not validate weaponIdx.
ReinitialiseWeaponVariables ()	Re-initialises variables related to weapons. Call after modifying weaponList.
RepairWeapon (int weaponIdx)	Repair the health of a weapon with the zero-based index in list of weapons. Will also set the heat level to 0. Can be useful if a weapon has burnt out after being over heated.
SetWeaponHeatLevel (int weaponIdx, float newHeatLevel)	Set the heat level of the Weapon with the zero-based index in list of weapons. Heat Level Range 0.0 (min) to 100.0 (max).
SetWeaponTarget (string weaponName, GameObject target)	Sets the gameobject that the weapon will attempt to aim at. Currently only applies to Weapon.WeaponType.TurretProjectile and FixedProjectile weapons with guided projectiles. However, for the sake of performance, this method does not do any WeaponType validation. WARNING: This method will generate garbage. Where possible call SetWeaponTarget(int weaponIdx, GameObject target). If only the name is known, first call GetWeaponIndexByName(string weaponName) once in your Awake() routine.
SetWeaponTarget (int weaponIdx, GameObject target)	Sets the gameobject that the weapon will attempt to aim at. Currently only applies to Weapon.WeaponType.TurretProjectile and FixedProjectile weapons with guided projectiles. However, for the sake of performance, this method does not do any WeaponType validation. WARNING: For the sake of performance, does not validate weaponIdx.

Property or Method	Description
SetWeaponTargetShip (int weaponIdx, ShipControlModule targetShipControlModule)	<p>Sets the ship that the weapon will attempt to aim at. Currently only applies to Weapon.WeaponType.TurretProjectile and FixedProjectile weapons with guided projectiles. However, for the sake of performance, this method does not do any WeaponType validation.</p> <p>WARNING: For the sake of performance, does not validate weaponIdx.</p>
SetWeaponTargetShipDamageRegion (int weaponIdx, ShipControlModule targetShipControlModule, DamageRegion damageRegion)	<p>Sets the ship's Localised damage region that the weapon will attempt to aim at. Currently only applies to Weapon.WeaponType.TurretProjectile and FixedProjectile weapons with guided projectiles. However, for the sake of performance, this method does not do any WeaponType validation. WARNING: For the sake of performance, does not validate weaponIdx.</p>
SetWeaponFireDirection (int weaponIdx, Vector3 aimAtWorldSpacePosition)	<p>Performs a once-off change to the direction the weapon will fire based on a "target" position in world-space. The weapon will NOT stay locked onto this position.</p> <p>NOTE: Should NOT be used for Turrets. Use SetWeaponTarget(..) instead.</p> <p>WARNING: For the sake of performance, does not validate weaponIdx.</p>
WeaponHasLineOfSight (Weapon weapon, bool directToTarget = false, bool obstaclesBlockLineOfSight = true, bool anyEnemy = true)	<p>Returns whether a weapon has line of sight to the weapon's specified target (i.e. weapon.target). If directToTarget is set to true, will raycast directly from the weapon to the target. If directToTarget is set to false, will raycast in the direction the weapon is facing. This method will return true if the raycast hits:</p> <ul style="list-style-type: none"> a) The weapon's specified target, b) An enemy ship (distinguished by faction ID) - even if it is not the target and anyEnemy is true c) An object that isn't the target (if obstaclesBlockLineOfSight is set to false), d) Nothing. <p>This method will return false if the raycast hits:</p> <ul style="list-style-type: none"> a) A friendly ship (distinguished by faction ID), b) An object that isn't the target (if obstaclesBlockLineOfSight is set to true). c) An enemy ship that is not the target when anyEnemy is false
WeaponHasLineOfSight (Weapon weapon, GameObject target, bool directToTarget = false, bool obstaclesBlockLineOfSight = true, bool anyEnemy = true)	<p>Returns whether a weapon has line of sight to a target. If directToTarget is set to true, will raycast directly from the weapon to the target. If directToTarget is set to false, will raycast in the direction the weapon is facing. This method will return true if the raycast hits:</p> <ul style="list-style-type: none"> a) The target, b) An enemy ship (distinguished by faction ID) - even if it is not the target and anyEnemy is true, c) An object that isn't the target (if obstaclesBlockLineOfSight is set to false), d) Nothing. <p>This method will return false if the raycast hits:</p> <ul style="list-style-type: none"> a) A friendly ship (distinguished by faction ID), b) An object that isn't the target (if obstaclesBlockLineOfSight is set to true).

Property or Method	Description
	c) An enemy ship that is not the target when anyEnemy is false

Ship API Call Backs

Custom runtime methods should be a lightweight to avoid performance issues. These single-cast delegates can have a single instance of a call-back method. This is useful when you want to take some (custom) action when something occurs, like when a ship is destroyed or is hit or damaged by a projectile.

These can be referenced or called from shipControlModule.shipInstance.

Property or Method	Description
CallbackOnDamage callbackOnDamage	The name of the custom method that is called immediately after damage has changed. Your method must take 1 float parameter. This should be a lightweight method to avoid performance issues. It could be used to update a HUD or take some other action.
CallbackOnCameraShake callbackOnCameraShake	Generally reserved for internal use by ShipCameraModule. If you use your own camera scripts, you can create a lightweight custom method and assign it at runtime so that this is called whenever camera shake data is available.
CallbackOnWeaponFired callbackOnWeaponFired	The name of the custom method that is called immediately after a weapon has fired. This should be a lightweight method to avoid performance issues that doesn't hold references the ship or weapon past the end of the current frame.
CallbackOnWeaponNoAmmo callbackOnWeaponNoAmmo	The name of the custom method that is called immediately after the weapon runs out of ammunition. Your method must take a ship and weapon parameter. This should be a lightweight method to avoid performance issues that doesn't hold references the ship or weapon past the end of the current frame.

Player Input Module Properties

Property or Method	Description
GetShipInput	Gets a reference to the input being sent from the PlayerInputModule to the ShipControlMode
GetShipControlModule	Get a reference to the ShipControlModule that this PlayerInputModule will send input data to.
IsInitialised	Is the Player Input Module initialised and ready for use?
IsInputEnabled	Is the PlayerInputModule currently enabled to send input to the ship? See EnableInput() and DisableInput(..)
IsCustomPlayerInputOnlyEnabled	Is all input except CustomerPlayerInputs ignored? See EnableInput() and DisableInput(..)
IsShipAIModeEnabled	Is the ship currently getting movement input from the ShipAllInputModule rather than PlayerInputModule?

Player Input Module (General) API Methods

Property or Method	Description
EnableInput ()	Enable the Player Input Module to receive input from the configured device
DisableInput (bool allowCustomPlayerInput = false)	Disable input or stop the Player Input Module from receiving input from the configured device. Also calls ResetInput(). When allowCustomPlayerInput is true, all other input except CustomPlayerInputs are ignored. This can be useful when you want to still receive actions that generally don't involve ship movement.
Initialise ()	Call this via code if "Initialise on Awake" is not enabled.
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.
ResetInput()	Reset and send 0 input on each axis to the ship
ReinitialiseCustomPlayerInput ()	This should be called if you modify the CustomPlayerInputs at runtime
ReinitialiseDiscardData ()	Re-initialise (set) the shipInput based on the is[axis/button]DataDiscard field settings. Must be called after each change to any of those fields/variables.
SetTargetDisplay (int displayNumber)	Used on a PC with multiple screens (display monitors) to adjust mouse position when player wants to use a screen other than Display 1.

Custom Player Inputs can be added at runtime. Here is a code fragment that demonstrates one scenario.

```
private void Start()
{
    // Do stuff here

    PlayerInputModule playerInputModule = GetComponent<PlayerInputModule>();

    // In code, add the input for the button press (assume Direct Keyboard)
    CustomPlayerInput customPlayerInput = new CustomPlayerInput();

    if (customPlayerInput != null)
    {
        customPlayerInput.inputAxisMode = PlayerInputModule.InputAxisMode.SingleAxis;
        customPlayerInput.isButton = true;
        customPlayerInput.isButtonEnabled = true;

        // Here we use Direct Keyboard but could also setup Unity Input System, Oculus, VIVE, Rewired etc.
        customPlayerInput.dkmPositiveKeycode = KeyCode.L;

        // Tell SSC to call our method when the button is pressed.
        customPlayerInput.customPlayerInputEvt = new CustomPlayerInputEvt();
    }
}
```

```

// Add a delegate listener. You could also call a method that takes
// a parameters of (Vector3 inputValue, int customPlayerInputEventType).
customPlayerInput.customPlayerInputEvt.AddListener(delegate { LaunchShip(); });

// Add the custom player input to the list in the PlayerInputModule
playerInputModule.customPlayerInputList.Add(customPlayerInput);

// We have modified the Custom Player Inputs, so we need to reinitilise them.
playerInputModule.ReinitialiseCustomPlayerInput();
}

// Do other stuff here
}

/// <summary>
/// Count down and then launch the ship using the catapult
/// </summary>
public void LaunchShip()
{
    if (isInitialised)
    {
        shipDisplayModule.ShowDisplayMessage(countdownDisplayMessage);

        // Count down with HUD messages
        Invoke(countDownMethodName, 1f);
        Invoke(countDownMethodName, 2f);
        Invoke(countDownMethodName, 3f);

        // Undock in CountDown() when countDown reaches 0.
    }
}

private void OnDestroy()
{
    if (isInitialised)
    {
        playerInputModule.RemoveListeners();
    }
}

```

Player Input Module (AI-Assist) API Methods

See also “Ship Docking Station API Methods” and “Ship Docking API Methods”.

Property or Method	Description
DisableAIMode ()	Attempt to disable AI control mode. This requires a Ship AI Input Module to also be attached to the player ship. After calling this method, check if it was successful with the IsShipAIModeEnabled property.
EnableAIMode ()	Attempt to enable AI control mode. This requires a Ship AI Input Module to also be attached to the player ship. After calling this method, check if it was successful with the IsShipAIModeEnabled property. When AI mode is enabled, the Custom Inputs are still enabled. To also disable Custom

Property or Method	Description
	Player Inputs, call <code>DisableInput(false)</code> after calling <code>EnableAIMode()</code> .
<code>EnableAIDocking ()</code>	Attempt to enter Ship AI-assisted mode, then attempt docking. This will leave and Custom Player Inputs enabled. To disable them, call <code>DisableInput(false)</code> after calling after calling <code>EnableAIDocking()</code> .
<code>EnableAIUndocking ()</code>	Attempt to enter Ship AI-assisted mode, then attempt undocking. This will leave and Custom Player Inputs enabled. To disable them, call <code>DisableInput(false)</code> after calling after calling <code>EnableAIUndocking()</code> .
<code>ToggleAIMode ()</code>	Attempt to switch between Player and AI control or vis versa. If the mode is changed during this method, we can assume that the Ship AI Input Module component is present and initialised.
<code>ToggleAIDocking ()</code>	Attempt to enter Ship AI-assisted mode, then attempt docking or undocking depending upon the current <code>DockignState</code> .

Player Input Module (XR) API Methods or Properties

This section contains a selection of useful properties and methods for use with Unity XR in a VR game or project. See also the Player Input Module chapter.

Property or Method	Description
<code>DisableXRCamera()</code>	Attempt to disable the XR Camera. Has no effect if UnityXR is not configured.
<code>DisableXRHands()</code>	Attempt to disable the XR Hands. Has no effect if UnityXR is not configured.
<code>EnableXRCamera()</code>	Attempt to enable the XR Camera. Has no effect if UnityXR is not configured.
<code>EnableXRHands()</code>	Attempt to enable the XR Hands. Has no effect if UnityXR is not configured.
<code>GetXRInputActionAsset()</code>	Get the Unity Input System <code>InputActionAsset</code> scriptableobject for XR.
<code>SetXRFirstPersonCamera1 (Camera newCamera, Transform cameraTrfm, bool isAutoEnable)</code>	Set the XR camera which will be rotated by the Tracked Pose Driver.
<code>UpdateUnityXRActions (bool showErrors = false)</code>	Used at runtime to convert string unique identifiers for actions (GUIDs) into Unity Input System <code>ActionInputs</code> to avoid looking them up and incurring GC overhead each Update. If actions are modified at runtime, call this method. Has no effect if Unity Input System package is not installed.

Ship AI Input Module API Methods or Properties

This section contains a selection of useful properties and methods for the AI module. Others can be found by looking in the scripts\ai\ShipAIInputModule.cs script.

Property or Method	Description
AIStateActionInfo GetCurrentStateAction ()	Returns an enumeration indicating what the current state action for this AI ship is.
AssignShipsToEvade (List<Ship> shipsToEvadeList)	Assigns a list of ships to evade for this ship, to be used by the current state.
AssignSurfaceTurretsToEvade(List<SurfaceTurretModule> surfaceTurretsToEvadeList)	Assigns a list of surface turrets to evade for this ship, to be used by the current state.
AssignTargetLocation(LocationData locationData)	Assigns a target location for this AI ship, to be used by the current state.
AssignTargetPath (PathData pathData)	Assigns a target path for this AI ship, to be used by the current state. Sets the current target path location index to 1 (the second point) or 0 (the first point) if there is no second point.
AssignTargetPath (PathData pathData, int previousPathLocationIndex, int nextPathLocationIndex, float targetPathTValue)	Assigns a target path for this AI ship, to be used by the current state. Set the previous and next locations along the path, and the normalised distance between the two locations where the ship will join the path.
AssignTargetAngularDistance (float targetAngularDistance)	Assigns a target angular distance for this ship, to be used by the current state.
AssignTargetDistance (float targetDistance)	Assigns a target distance for this ship, to be used by the current state.
AssignTargetTime (float targetTime)	Assigns a target time for this ship, to be used by the current state.
AssignTargetPosition (Vector3 targetPositionVector)	Assigns a target position for this AI ship, to be used by the current state.
AssignTargetRadius(float targetRadius)	Assigns a target radius for this ship, to be used by the current state.
AssignTargetRotation (Quaternion targetRotationQuaternion)	Assigns a target rotation for this AI ship, to be used by the current state.
AssignTargetShip (ShipControlModule targetShipControlModule)	Assigns a target ship for this AI ship, to be used by the current state.
AssignTargetVelocity (Vector3 targetVelocity)	Assigns a target velocity for this ship, to be used by the current state.
DisableAI()	Prevent the main update loop from performing calculations or sending data to the ship. This is different from setting the state to AIState.idleStateID.
EnableAI()	Enable the main update loop to perform calculations and send input to the ship.

Property or Method	Description
GetCurrentTargetPathTValue()	Get the time value between the previous Target Path Location and the current (next) Location along the Path. The TValue should be between 0.0 and 1.0.
GetLastBehaviourInputTarget ()	Returns the last position to be designated as the target position by the chosen AI behaviour input.
GetShipControlModule	When the Ship AI Input Module is initialised, this property returns the attached ShipControlModule without having to call GetComponent().
GetState()	Returns the state ID of the current state for this AI ship. Returns -1 if the currentState is not set. To get the instance of the state call AIState.GetState(shipAIInputModule.GetState())
GetCurrentStateStageIndex()	Gets the current stage index for the current state. This (zero-based) index is used to keep track of what stage the AI ship has reached in the current state.
GetCurrentTargetPathLocationIndex()	Gets the current index of the location of the target path the AI ship will head towards.
GetPreviousTargetPathLocationIndex()	Gets the previous index of the location of the target path the AI ship is heading away from.
GetShip	Get a reference to the Ship instance which is part of an initialised ShipControlModule. If the Ship AI Input Module or Ship Control Module are not initialised, it will return null.
GetShipId	Get the identity of the ship this AI module is attached to. It will return 0 if the ship is not initialised.
GetShipsToEvade ()	Gets the currently assigned list of ships to evade (if any).
GetSurfaceTurretsToEvade()	Gets the currently assigned list of surface turrets to evade (if any).
GetTargetAngularDistance()	Gets the currently assigned target angular distance.
GetTargetDistance()	Gets the currently assigned target distance.
GetTargetRadius()	Gets the currently assigned target radius.
GetTargetPosition()	Gets the currently assigned target position (if any). A returned value of Vector3.Zero indicates it is unassigned.
GetTargetLocation()	Gets the currently assigned target location (if any).
GetTargetPath()	Gets the currently assigned target path (if any).
GetTargetRotation()	Gets the currently assigned target rotation (if any).
GetTargetTime()	Gets the currently assigned target time.
GetTargetVelocity()	Gets the currently assigned target velocity.

Property or Method	Description
HasCompletedStateAction ()	Returns whether the state action has been completed yet.
Initialise ()	Initialises the Ship AI Input Module.
IsAIEnabled	Can the main update loop perform calculations and send input to the ship?
IsInitialised	Has the AI module been initialised?
RecalculateShipParameters ()	Recalculates the parameters for the AI's "model" of the ship. Should be called if any of the ship's characteristics are modified.
ResetPIDControllers ()	Resets the ship's PID Controllers. Call this if you manually modify the ship's velocity or angular velocity.
SetCurrentStateStageIndex(int newStateStageIndex)	Sets the current stage index for the current state. This (zero-based) index is used to keep track of what stage the AI ship has reached in the current state. Typically, this should only be set from inside a state method.
SetCurrentTargetPathLocationIndex (int newTargetPathLocationIndex)	Sets the current index of the location of the target path the AI ship will head towards. If the index value has changed, also updates the Previous Target Path Location Index.
SetCurrentTargetPathLocationIndex (int newTargetPathLocationIndex, float newTargetPathLocationTValue)	Sets the current index of the location of the target path the AI ship will head towards. If the index value has changed, also update the Previous Target Path Location Index. Also sets the time value between the previous location and the current (next) location. The time value should be between 0.0 and 1.0
SetCurrentTargetPathTValue(float tValue)	Set the time value between the previous Target Path Location and the current (next) Location along the Path. The TValue should be between 0.0 and 1.0.
SetHasCompletedStateAction (bool isCompleted = true)	Set whether the state action has been completed yet. Typically, this should only be called from within a state method.
SetPreviousTargetPathLocationIndex(int newTargetPathLocationIndex)	Sets the previous index of the location of the target path the AI ship is heading away from.
SetState (int newStateID)	Sets the current state for this AI ship using the given state ID.
Ship GetTargetShip()	Gets the currently assigned target ship (if any).
TelePort (Vector3 delta, bool resetVelocity)	Teleport the AI ship to a new location by moving by an amount in the x, y and z directions. This could be useful if changing the origin or centre of your world to compensate for float-point error. It could also be used it as part of a custom hyper-drive system you have developed.

Ship AI Input Module API Call Backs

Custom runtime methods should be a lightweight to avoid performance issues. These single-cast delegates can have a single instance of a call-back method. This is useful when you want to take some (custom) action when something occurs, like when a behaviour output is set, and action has been completed, or when the AIState is changed.

See Demos\Scripts\DemoFlyToLocation.cs for an example of writing a custom behaviour.

These can be referenced or called from an instance of shipAIInputModule.

Property or Method	Description
CallbackCompletedStateAction callbackCompletedStateAction	The name of the developer-supplied custom method that is called when the current state action has been completed. For example, a ship has got to the end of a path. Must have 1 parameter of type ShipAIInputModule.
CallbackCustomDockBehaviour callbackCustomDockBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomDock".
CallbackCustomEvasionBehaviour callbackCustomEvasionBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomEvasion".
CallbackCustomFleeBehaviour callbackCustomFleeBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomFlee".
CallbackCustomFollowPathBehaviour callbackCustomFollowPathBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomFollowPath".
CallbackCustomIdleBehaviour callbackCustomIdleBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomIdle".
CallbackCustomObstacleAvoidanceBehaviour callbackCustomObstacleAvoidanceBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomObstacleAvoidance".
CallbackCustomPursuitArrivalBehaviour callbackCustomPursuitArrivalBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomPursuitArrival".
CallbackCustomPursuitBehaviour callbackCustomPursuitBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomPursuit".
CallbackCustomSeekArrivalBehaviour callbackCustomSeekArrivalBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomSeekArrival".
CallbackCustomSeekBehaviour callbackCustomSeekBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomSeek".
CallbackCustomSeekMovingArrivalBehaviour callbackCustomSeekMovingArrivalBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomSeekMovingArrival".
CallbackCustomUnblockConeBehaviour callbackCustomUnblockConeBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomUnblockCone".
CallbackCustomUnblockCylinderBehaviour callbackCustomUnblockCylinderBehaviour	The name of the developer-supplied custom method that is called when the AIBehaviourType is "CustomUnblockCylinder".
CallbackOnStateChange callbackOnStateChange	<p>The name of the developer-supplied custom method that gets called whenever the state changes. Must have 3 parameters of type: ShipAIInputModule, int (currentStatId), and int (previousStatId).</p> <p>You can get the actual AIState by calling say AIState.GetState (previousStatId).</p>

Ship Camera Module API Methods

Property or Method	Description
ApplyCameraSettings (ShipCameraSettings shipCameraSettings)	Attempt to apply camera settings from a ScriptableObject to this ShipCameraModule. See also ExportCameraSettings(..)
DisableCamera()	Disables the camera, preventing it from moving. It does not stop the camera from rendering. If you set Time.timeScale to 0 in your game, you must first call this method. Typically, you'll also want to call MoveCamera() just before DisableCamera() for the current frame. See also StopCamera().
DisableZoom()	Turn off the zoom feature and reset the camera field of view to the default setting.
EnableCamera()	Enables the camera, allowing it to move and follow or aim at a target ship. See StartCamera() to allow it to render.
EnableZoom()	Turn on the zoom feature
ExportCameraSettings (ref ShipCameraSettings shipCameraSettings)	Attempt to export the camera settings from this ShipCameraModule to a pre-created ShipCameraSettings ScriptableObject. See also ApplyCameraSettings(..)
GetCamera()	Get the camera being used by this module
GetTarget ()	Get the current target ship for the camera. If the camera is currently not assigned to a ship, it will return null.
GetZoomAmount()	The amount of zoom currently used. 0 is no zoom. 1 is fully zoomed in. -1 is fully zoomed out.
IsCameraInFixedUpdate	Is the camera being moved using FixedUpdate()?
IsCameraStarted	Is camera started and rendering?
IsCameraEnabled	Is the camera enabled for movement?
MoveCamera()	Typically, you should not call this yourself, as it is called automatically each frame. However, it can prove useful in the case where you need to force a camera movement update for this frame.
MoveTo(Vector3 position, Vector3 rotation)	Tele-port the camera to a new position. Rotation is xyz euler angles in degrees. See also TelePort(..)
ReinitialiseTargetVariables ()	Re-initialises variables related to the target. Call after modifying the camera "target" public variable.
SendZoomInput (float zoomInput)	Send zoom input to the camera. It is ignored if Zoom is not enabled. Values should be between -1.0 and 1.0. Zoom in for +ve values. Zoom out for -ve values. The value is automatically reset to 0 after use.
SetTarget (ShipControlModule shipControlModule)	Set's a new target for the camera and calls ReinitialiseTargetVariables().
SetUnzoomDelay (float newValue)	Set the unzoom delay, in seconds, for the camera.

Property or Method	Description
SetZoomAmount (float zoomAmount)	The amount of zoom to apply. 0 is no zoom. 1 is fully zoomed in. -1 is fully zoomed out.
ShakeCamera()	Shake the camera for maxShakeDuration seconds which maxShakeStrength or force. If the camera is not enabled or the duration and/or strength are 0 or less, StopCameraShake() will be automatically called and the inputs ignored.
ShakeCamera (float relativeStrength)	Shake the camera with strength and duration relative to the current maxShakeDuration and maxShakeStrength. Range of relativeStrength should be between 0.0 and 1.0.
ShakeCamera (float duration, float strength)	Shake the camera for specified seconds which the given relative strength or force. If the camera is not enabled or the duration and/or strength are 0 or less, StopCameraShake() will be automatically called and the inputs ignored.
ShakeCameraDelayed (float delayTime)	Shake the camera after initial delay in seconds, with the current maxShakeDuration and maxShakeStrength.
StartCamera ()	Start the camera rendering. If the camera is disabled, it will remain so. Call EnableCamera() to allow it to also move. If there was an Audio Listener component attached and enabled when this module was initialised, the listener will be re-enabled.
StopCamera ()	Stops the camera from rendering. Also disables the camera from moving. If there was an Audio Listener component attached and enabled when this module was initialised, the listener will be disabled. See also DisableCamera().
StopCameraShake()	Stop the camera from shaking
SetCameraTargetDisplay (int displayNumber)	Set the camera to use a particular display or screen. Displays or monitors are numbered from 1 to 8.
TelePort (Vector3 delta)	Teleport the camera to a new location by moving by an amount in the x, y and z directions. This could be useful if changing the origin or centre of your world to compensate for float-point error. It could also be used it as part of a custom hyper-drive system you have developed.
TelePort (Vector3 newPosition, Quaternion newRotation)	Teleport the camera to a new location with given rotation in world space.

Auto Targeting Module API Methods or Properties

Property or Method	Description
IsInitialised	Is the AutoTargetingModule initialised and ready for use? If not, call Initialise() or set initialiseOnStart in the Inspector.
IsTargetsShownOnHUD	Are the radar targets being sent to the HUD?
NumberOfTargetsInRange	The number of targets that are currently in range of weapons with auto-targeting enabled.

Property or Method	Description
SetHUD (ShipDisplayModule shipDisplayModule)	If the Module Mode is ShipControlModule, assign the HUD to the AutoTargetingModule - else set it to null.
ShowTargetsOnHUD ()	If the HUD is initialised and shown, start sending Target data to the HUD. Only Display Targets already on the HUD will be updated. See also shipDisplayModule.AddTarget(..)
HideTargetsOnHUD ()	Stop sending targeting data to the HUD. Also, turn off any Display Targets on the HUD.

Auto Targeting Module API Call Backs

Callback	Description
CallbackOnTargeted1 callbackOnTargeted1	<p>The name of the custom method that is called immediately after the ship weapon acquires a new target. Your method must take 4 parameters (ShipControlModule, ShipControlModule, DamageRegion and Gameobject). Any of the last 3 parameters can be null. This should be a lightweight method to avoid performance issues.</p> <pre>autoTargetingModule callbackOnTargeted1= YourMethod;</pre> <pre>public void YourMethod (ShipControlModule sourceShip, ShipControlModule targetShip, DamageRegion targetDamageRegion, GameObject targetObject) { // Your code here }</pre>
CallbackOnTargeted2 callbackOnTargeted2	<p>The name of the custom method that is called immediately after the ship weapon acquires a new target. Your method must take 4 parameters (SurfaceTurretModule, ShipControlModule, DamageRegion and Gameobject). Any of the last 3 parameters can be null. This should be a lightweight method to avoid performance issues.</p> <pre>autoTargetingModule callbackOnTargeted1= YourMethod;</pre> <pre>public void YourMethod (SurfaceTurretModule sourceSurfaceTurret, ShipControlModule targetShip, DamageRegion targetDamageRegion, GameObject targetObject) { // Your code here }</pre>

Location and Path API Methods

Property or Method	Description
SSCManager.GetOrCreateManager (int sceneHandle = 0)	Static method to get a reference to the Sci-Fi Ship Controller Manager instance in the scene. The SSCManager manages Locations and Paths in the scene. For multi-additive scenes, pass in the current scene handle e.g., gameObject.scene.handle.
SSCManager.AddLocation (LocationData locationData)	Add a new Location in the scene. The Location can then be added to one or more Paths. Locations can be used with SSC AI and can appear on radar.
SSCManager.AddLocation (Vector3 wsPosition, bool autoGenerateName)	Same as calling AppendLocation(wsPosition, autoGenerateName)
SSCManager.GetNextLocation (PathData pathData, int currentIdx, bool isWrapEnabled)	Static method to get the next Location that has been assigned to a Path, based on the 0-based index position in list of PathLocationData items. May return null if no next found. Wraps to start if isWrapEnabled is true. If currentIdx = -1, it will attempt to find the first assigned Location. e.g. LocationData locationData = GetNextLocation(pathData, prevIdx, false);
SSCManager.GetNextPathLocationIndex (PathData pathData, int currentIdx, bool isWrapEnabled)	Static method to get the next PathLocation index that has been assigned to a Path, based on the 0-based index position in list of PathLocationData items. May return -1 if no next found. Wraps to start if isWrapEnabled is true. If currentIdx = -1, it will attempt to find the first assigned Location.
SSCManager.GetPreviousLocation (PathData pathData, int currentIdx, bool isWrapEnabled)	Static method to get the previous Location that has been assigned to a Path, based on the 0-based index position in list of PathLocationData items. May return null if no previous found. Wraps to end if isWrapEnabled is true. If currentIdx == number of Locations in Path, the last assigned Location will be returned.
SSCManager.GetPreviousPathLocationIndex (PathData pathData, int currentIdx, bool isWrapEnabled)	Static method to get the previous PathLocation index that has been assigned to a Path, based on the 0-based index position in list of PathLocationData items. May return -1 if no previous found. Wraps to end if isWrapEnabled is true. If currentIdx == number of Locations in Path, the last assigned Location will be returned.
sscManager.AppendLocation (Vector3 wsPosition, bool autoGenerateName)	Add a new Location in the scene. The Location can then be added to one or more Paths. Locations can be used with SSC AI. If autoGenerateName is true, GC will be impacted.
sscManager.AppendLocation (PathData pathData, Vector3 wsPosition, bool autoGenerateName, bool refreshPath)	Add a new Location in the scene and add it to the Path. If autoGenerateName is true, GC will be impacted. refreshPath will update the path distances. If adding many Locations, set refreshPath to false and call RefreshPathDistances(..) for each Path manually.
sscManager.AddPath (PathData newPath)	Add a path to the scene. Alternatively, use CreatePath().

Property or Method	Description
<code>sscManager.CreatePath()</code>	Create a new path and return the reference to the PathData.
<code>sscManager.InsertLocationAt (PathData pathData, int insertIndex, bool autoGenerateName, bool refreshPath)</code>	Insert a new Location before the zero-based point on the Path. It has no effect if there are not no points in the Path.
<code>sscManager.UpdateLocation (LocationData locationData, Vector3 wsPosition, bool refreshPath)</code>	Update the Location's position. This also updates tangent control points for any Paths that the Location is a member of. <code>refreshPath</code> will update the path distances for all affected Paths. If updating many Locations, set <code>refreshPath</code> to false and call <code>RefreshPathDistances(..)</code> for each Path manually. If enabled for radar, update the radar item.
<code>sscManager.InitialiseLocations (PathData pathData, Vector3 dockingStationPosition, Vector3 offsetPosition, Quaternion deltaRotation, Vector3 pathVelocity, Vector3 pathAngularVelocity)</code>	Set the initial or starting positions of Locations and control points along a dynamic (moveable) Path at runtime. Currently used with ShipDockingStation entry and exit paths that are attached to a mothership. NOTE: Assumes the Locations in each Path are not members of any other Paths in the scene. Paths will only be initialised once.
<code>sscManager.MoveLocations (PathData pathData, LocationData locationDataMoved, Vector3 wsPosition, bool refreshPath)</code>	Given a Location in a Path, being moved, update all other selected Locations in the same Path at the same time. NOTE: As this uses delegates, it may incur some GC overhead.
<code>sscManager.MoveLocations (PathData pathData, float updateSeqNumber, Vector3 deltaPosition, Quaternion deltaRotation)</code>	Designed specifically for runtime movement of a Path, it assumes the Locations of the Path are NOT members of any other Path. Position and Rotation deltas are from the CURRENT positions and rotations. Works similar to TelePort API methods.
<code>sscManager.MoveLocations (PathData pathData, float updateSeqNumber, Vector3 dockingStationPosition, Vector3 deltaPosition, Quaternion deltaRotation, Vector3 pathVelocity, Vector3 pathAngularVelocity)</code>	Designed specifically for runtime movement of a Path, it assumes the Locations of the Path are NOT members of any other Path. Position and Rotation deltas are from the INITIAL or starting positions and rotations. See also <code>InitialiseLocations(..)</code> .
<code>sscManager.MoveLocations (LocationData locationDataMoved, Vector3 wsPosition, bool refreshPaths)</code>	Given a Location being moved, update all other selected Locations in the scene at the same time. If they are members of Paths, update those Paths.
<code>sscManager.DeleteSelectedLocations (bool refreshPath)</code>	Delete all the selected Locations, including any Location slots associated with a Path. <code>refreshPath</code> will update the path distances for all affected Paths.
<code>sscManager.GetLocation (int guidHash)</code>	Get a Location given the unique guidHash code of the Location
<code>sscManager.GetLocation (string locationName)</code>	Get the first Location given the name of the Location. <code>locationName</code> is not case sensitive. If the <code>locationName</code> parameter is empty or null, no Location will be returned even if there are Locations without a name. NOTE: When possible, use <code>GetLocation(guidHash)</code> .

Property or Method	Description
<code>GetLocationByGameObjectID</code> (int gameObjectInstanceID)	If a location was added with <code>AppendLocation(gameObject)</code> , it can also be retrieved at runtime with this method by passing in the <code>gameObject.GetInstanceID()</code> . This might be useful if you wish to destroy the <code>gameObject</code> (and remove the <code>Location</code>) but don't have the <code>guidHash</code> or <code>LocationData</code> reference.
<code>sscManager.GetPath</code> (int guidHash)	Get a Path given the unique <code>guidHash</code> code of the Path
<code>sscManager.GetPath</code> (string pathName)	Get the first Path given the name of the Path. <code>pathName</code> is not case sensitive. If the <code>pathName</code> parameter is empty or null, no Path will be returned even if there are Paths without a name. NOTE: When possible, use <code>GetPath(guidHash)</code> .
<code>sscManager.GetPathDistance</code> (PathData pathData, int pathLocationIndex1, int pathLocationIndex2)	Get the Path distance between two Locations. If the Location index positions in the Path are invalid, the method will return 0. If <code>pathLocationIndex2</code> is less than <code>pathLocationIndex1</code> , assume the path wraps around to the start again.
<code>sscManager.RefreshPathDistances</code> (PathData pathData)	Recalculate the distances that are stored in the <code>PathLocationData</code> instances.
<code>sscManager.ReversePath</code> (PathData pathData)	Attempt to reverse the direction of a path.
<code>sscManager.SnapPathToMesh</code> (testPath, Vector3.up, ~0, true);	Attempt to snap the Path locations to a mesh above or below the current location. The location is placed with an offset from mesh. This 'snap' distance is the <code>pathData.locationDefaultNormalOffset</code> . The method checks for a mesh between <code>pathData.snapMinHeight</code> and <code>snapMaxHeight</code> in the <code>upDirection</code> .
<code>sscManager = SSCManager.GetOrCreateManager();</code> <code>LocationData locn =</code> <code>sscManager.GetLocation("MyLocation");</code> <code>sscManager.SnapLocationToMesh</code> (locn, 10f, 0f, 2000f, Vector3.up, ~0, true);	Attempt to snap the Location to a mesh above or below the current position. The location is placed with an offset from mesh. This 'snap' distance is the offset.

Beam, Destruct, Projectile and Effects API Methods

Beams, Destruct, Projectiles, and Effects objects are instantiated and managed by the `SSCManager` within a scene. Properties and methods are accessible via an instance of `sscManager` - see `GetOrCreateManager()` below.

Property or Method	Description
<code>bool IsBeamsPaused</code>	[READONLY] Are all the beam objects in the scene currently paused?
<code>bool IsEffectsObjectsPaused</code>	[READONLY] Are all the effects objects in the scene currently paused?
<code>bool IsProjectilesPaused</code>	[READONLY] Are all the projectiles in the scene currently paused?

Property or Method	Description
CreateEffectsPools (SSCSoundFXSet sscEffectsSet, int[] effectsPrefabIDs)	Create any Effects pools that have not already been created using a set of EffectsModule prefabs. Populate a pre-created array of EffectsTemplate prefabIDs. The length of effectsPrefabIDs must match number of effects slots.
GetBeamPrefab (int beamPrefabID)	Returns the prefab for a beam given its beam prefab ID.
GetEffectsObjectPrefab (int effectsObjectPrefabID)	Returns the prefab for an EffectsObject given its effects prefab ID.
GetProjectilePrefab (int projectilePrefabID)	Returns the prefab for a projectile given its projectile prefab ID.
GetOrCreateEffectsPool (EffectsModule effectsObjectPrefab)	Get the effects pool for this prefab or create a new pool if one does not already exist. Return the effectsObjectPrefabID for the pool. See also InstantiateEffectsObject(ieParms), and InstantiateSoundFX(..).
InstantiateEffectsObject (ref InstantiateEffectsObjectParameters ieParms)	Instantiates the effects object with ID effectsObjectPrefabID at the position specified in world space (effectsObjectPrefabID is the ID sent back to each projectile / damage region after calling the UpdateProjectilesAndEffects() method).
InstantiateSoundFX (InstantiateSoundFXParameters sfxParms, AudioClip audioClip)	Instantiate a pooled EffectsModule which contains an audio source. If audioClip is null, the existing one (if there is one) attached to the prefab will be used. See also GetOrCreateEffectsPool(..).
PauseBeams()	Pause all Pooled, and Non-Pooled Beams in the scene. This can be useful if you want to pause a game or bring up menu.
PauseDestructs()	Pause all Pooled, and Non-Pooled Destruct objects in the scene. This can be useful if you want to pause a game or bring up menu.
PauseEffectsObjects()	Pause all Pooled and Non-Pooled Effects Objects in the scene. This can be useful if you want to pause a game or bring up menu.
PauseProjectiles()	Pause all DOTs, Pooled, and Non-Pooled Projectiles in the scene. This can be useful if you want to pause a game or bring up menu.
PlaySoundFX (int sfxObjectPrefabID, AudioClip audioClip, Vector3 audioPosition, float clipVolume)	Play an audio clip at the specified world-space position at a volume. This uses the pooled EffectsModules created with GetOrCreateEffectsPool(..). See also InstantiateSoundFX(..).
ResumeBeams ()	Resume all the Pooled and Non-Pooled Beams in the scene.
ResumeDestructs()	Resume all the Pooled and Non-Pooled Destruct objects in the scene.
ResumeEffectsObjects()	Resume all Pooled and Non-Pooled Effects Objects in the scene.
ResumeProjectiles()	Resume all DOTs, Pooled, and Non-Pooled Projectiles in the scene. This can be useful if the game was previously paused.

Property or Method	Description
<code>SSCManager.GetOrCreateManager()</code>	Static method to get a reference to the Sci-Fi Ship Controller Manager instance in the scene. The <code>SSCManager</code> is what manages projectiles and effects in the scene.
<code>TelePortBeams (Vector3 delta)</code>	Teleport or instantly move all active (weapon) beams by an amount in the x, y and z directions. This could be useful if changing the origin or centre of your world to compensate for float-point error.
<code>TelePortProjectiles (Vector3 delta)</code>	Teleport or instantly move all active projectiles by an amount in the x, y and z directions. This could be useful if changing the origin or centre of your world to compensate for float-point error.

Projectile API Call Backs

Projectile callback can be useful when you wish to say write your own guided projectile algorithm rather than use the one built-in.

Callback	Description
<code>CallbackProjectileMoveTo</code> <code>callbackProjectileMoveTo</code>	<p>The name of the custom method that is called when a kinematic guided projectile is being moved. Your method must take 1 parameter of type <code>ProjectileModule</code>. It MUST correctly update the <code>Velocity</code> property AND <code>transform.position</code> on the projectile. Optionally it can update <code>transform.rotation</code>. This should be a lightweight method to avoid performance issues as it is typically called each <code>FixedUpdate</code>.</p> <p><code>SSCManager.GetOrCreateManager().callbackProjectileMoveTo = YourMethod;</code></p> <pre> public void YourMethod (ProjectileModule projectileModule) { // Your code here } </pre>

Destructible Object Module API Methods or Properties

This module can be used to trigger a `DestructModule` when the health of the object reaches 0.

Property or Method	Description
<code>float Health</code>	The current health value of this object.
<code>float HealthNormalised</code>	Normalised (0.0 – 1.0) value of the health of the object.
<code>bool IsInitialised</code>	Has the module been initialised?
<code>int RadarId</code>	The number used by the <code>SSCRadar</code> system to identify this object at a point in time. This should not be stored across frames and is updated as required by the system.
<code>float ShieldHealth</code>	The current health value of this object's shield. When a shield is destroyed, its value is set to -0.01.

Property or Method	Description
float ShieldNormalised	Normalised (0.0 – 1.0) value of the shield for this object. If useShielding is false, it will always return 0.
ApplyDamage (float damageAmount, ProjectileModule.DamageType damageType = ProjectileModule.DamageType.Default)	Apply damage to the object. If Use Damage Multipliers is enabled, you can optionally pass in the DamageTye.
ApplyDamage (CallbackOnObjectHitParameters callbackOnObjectHitParameters)	This routine is called by our damage receiver when a projectile or beam hits our object
AddHealth (float healthAmount, bool isAffectShield)	Add health to the object. If isAffectShield is true, and the health reaches the maximum configured, excess health will be applied to the shield. To incur damage, use the ApplyDamage(..).
DisableRadar()	Disable radar for this object. If you want to change the visibility to other radar consumers, consider calling SetRadarVisibility(..) rather than disabling the radar and (later) calling EnableRadar() again. This will be automatically called by DestructObject().
EnableRadar()	Enable radar for this object. It will be visible on radar to others in the scene.
GetDamageMultiplier (ProjectileModule.DamageType damageType)	Returns the damage multiplier for damageType.
ReinitialiseDestructObjects()	Reinitialises variables required for Destruct Module. Call after modifying the destructObject.
ReinitialiseEffects()	Reinitialises variables required for effects of the Destructible Object Module. Call after modifying any effect data for this module.
ResetHealth()	Reset the health of the object back to initial values
SetDamageMultiplier (ProjectileModule.DamageType damageType, float damageMultiplier)	Sets the damage multiplier for damageType to damageMultiplier.
SetRadarVisibility (bool isVisible)	If radar is enabled for this object, set its visibility to radar.
SetSquadronId (int newSquadronId)	Set the Squadron Id for the object. If radar is enabled, this will also update the radar.
SetFactionId (int newFactionId)	Set the Faction Id for the object. If radar is enabled, this will also update the radar.
VerifyMultiplierArray()	Verify that the damage multiplier array is correctly sized

The following methods can be overridden.

Property or Method	Description
virtual void Initialise()	Initialises the DestructibleObjectModule. If you wish to override this in a child (inherited) class you almost always will want to call the base method first. public override void Initialise()

Property or Method	Description
	<pre>{ base.Initialise(); // Do stuff here }</pre>
virtual void DestructObject()	<p>Destroys the object. If you wish to override this in a child (inherited) class you almost always will want to call the base after doing your actions.</p> <pre>public override void DestructObject() { // Do stuff here base.DestructObject(); }</pre>
virtual void DestructObjectDelayed (float delayDuration)	Destroys the object after a delay period in seconds

Radar API Methods or Properties

A good way to learn about the Radar API is to look at any sample or demo code in the asset or discuss your scenario with us in the Unity forum or on Discord.

As a general rule, if you see a parameter called “itemIndex”, for ships and locations this the RadarId. It is accessible via:

shipControlModule.shipInstance.RadarId or locationData.RadarId.

To get the instance in the scene of the radar system use the following code:

```
SSCRadar sscRadar = SSCRadar.GetOrCreateRadar();
```

Property or Method	Description
AddItem (SSCRadarItem sscRadarItemInput)	Add an item to be tracked by radar. You would typically use this for items created outside Sci-Fi Ship Controller. Ships and Locations automatically call this when EnableRadar() is called via the API or “Visible to Radar” is enabled in the editor. For your own objects, you’ll need to regularly call UpdateItem(..).
DisableRadar(int itemIndex)	The item will no longer be visible in the radar system. If you want to change the visibility to other radar consumers, consider changing the radar item data rather than disabling the radar and (later) calling EnableRadar(..) again.
EnableRadar(GameObject gameObjectToAdd, Vector3 position, int factionId, int squadronId, int guidHash, int blipSize)	Attempt to add any gameobject to the radar system. If added, it will be immediately visible to radar and the RadarId (itemIndex) will be return. A value of -1 indicates it wasn't added. NOTE: To add a ship to radar, use shipControlModule.EnableRadar(). For Locations use sscManager.EnableRadar(location).

Property or Method	Description
	If you set guidHash to 0, we will automatically set it to <code>gameObjectToAdd.GetHashCode()</code> .
<code>FollowGameObject</code> (<code>GameObject gameobject</code>)	The centre of the radar will follow the gameobject. Currently this only works when the built-in UI is configured in the editor under Visuals. The same can be achieved via the API when calling <code>GetRadarResults(..)</code> and passing in the transform position.
<code>FollowShip</code> (<code>ShipControlModule shipControlModule</code>)	The centre of the radar will follow the ship. Currently this only works when the built-in UI is configured in the editor under Visuals. The same can be achieved via the API when calling <code>GetRadarResults(..)</code> and passing in the ship <code>TransformPosition</code> .
<code>GetNextBlipInScreenViewPort</code> (<code>List<SSCRadarBlip> blipList</code> , <code>int startIndex</code> , <code>Camera camera</code> , <code>Vector2 viewSize</code> , <code>Vector2 viewportSize</code> , <code>Vector2 viewportOffset</code>)	Get the next blip in the supplied list, that is in front of the camera within a custom 2D viewport. The <code>startIndex</code> is the zero-based index to begin searching the supplied list. If a match is found, the zero-based index of that blip will be returned, else -1 will be returned. <code>viewSize</code> is usually the screen resolution. <code>viewportSize</code> is the width and height as 0.0-1.0 values of the full <code>viewSize</code> . <code>viewportOffset</code> is -1.0 to 1.0 values with 0,0 as the centre of the screen.
<code>GetNextBlipInView</code> (<code>List<SSCRadarBlip> blipList</code> , <code>int startIndex</code> , <code>Camera camera</code> , <code>Vector2 viewSize</code>)	Get the next blip in the supplied list, that is within the camera's viewing area. The <code>startIndex</code> is the zero-based index to begin searching the supplied list. If a match is found, the zero-based index of that blip will be returned, else -1 will be returned. <code>viewSize</code> is usually the screen resolution. NOTE: Currently assumes the camera is full screen.
<code>GetRadarItem</code> (<code>int itemIndex</code> , <code>int sequenceNumber</code>)	Get an active (assigned) radar item given the <code>itemIndex</code> AND its <code>sequenceNumber</code> . The sequence number is used to validate that this is the correct radar item. For ships and locations, the <code>itemIndex</code> is stored as <code>RadarId</code> . e.g. <code>shipControlModule.shipInstance.RadarId</code>
<code>GetRadarItem</code> (<code>SSCRadarItemKey sscRadarItemKey</code>)	Get an active (assigned) radar item given the <code>SSCRadarItemKey</code> . For ships and locations, the <code>sscRadarItemKey.radarItemIndex</code> is stored as <code>RadarId</code> . e.g. <code>shipControlModule.shipInstance.RadarId</code> .
<code>GetRadarItemIndexByHash</code> (<code>int guidHash</code>)	Location <code>radarItemTypes</code> store a unique <code>guidHash</code> . <code>GameObject radarItemTypes</code> by default store a Unity hash code. Ship Damage Regions <code>radarItemTypes</code> store the damage region unique <code>guidHash</code> . For these three types, when the <code>itemIndex</code> (or <code>RadarId</code>) is unknown, this method can be used to retrieve it. If there are no matches, -1 is returned.
<code>GetRadarResults</code> (<code>SSCRadarQuery sscRadarQuery</code> , <code>List<SSCRadarBlip> queryResultsList</code>)	Send a query (see below) to the radar system and populate a list of matching items. The <code>queryResultsList</code> must be a non-null, empty list. A query will never return items with <code>isVisibleToRadar = false</code> .

Property or Method	Description
GetVisibility (int itemIndex)	Does this item appear in Radar queries? If the itemIndex is invalid, this method will always return false. NOTE: For performance reasons, it doesn't check if this is non-empty radar item. For ships and locations, the itemIndex is stored as RadarId. e.g. shipControlModule.shipInstance.RadarId
HideUI()	If the UI (mini-map) has been configured in the Editor under Visuals, hide the radar display.
IsBlipInScreenViewPort (SSCRadarBlip radarBlip, Camera camera, Vector2 viewSize, Vector2 viewportSize, Vector2 viewportOffset)	Is the blip in front of the camera within a custom 2D viewport. Offset is -1.0 to 1.0 with 0,0 as the centre of the screen. viewportSize is 0.0-1.0 of viewSize width and height.
IsBlipInView (SSCRadarBlip radarBlip, Camera camera, Vector2 viewSize)	Is the blip within the camera's viewing area? viewSize is usually the screen resolution. NOTE: Currently assumes the camera is full screen.
IsGameObjectBlip (SSCRadarBlip radarBlip)	Does the radar blip contain information about a GameObject?
IsLocationBlip (SSCRadarBlip radarBlip)	Does the radar blip contain information about a Location?
IsShipBlip (SSCRadarBlip radarBlip)	Does the radar blip contain information about a ship?
IsShipDamageRegionBlip (SSCRadarBlip radarBlip)	Does the radar blip contain information about a ship damage region?
RefreshRadarImageStatus()	Call at runtime if radarImage is swapped or made null.
SetBlipSize (int itemIndex, byte blipSize)	Set the blip size of the radarItem. Typically, this is set automatically when a ship or Location is enabled for radar. However, this lets the blip size be overridden at runtime. NOTE: For performance reasons, it doesn't check if this is non-empty radar item. For ships, the itemIndex is stored as RadarId. e.g. shipControlModule.shipInstance.RadarId. For surfaceTurretModule, the itemIndex is stored as RadarId.
SetCanvasSortOrder (int newSortOrder)	Set the sort order in the scene of the radar mini-map. Higher values appear on top.
SetCanvasTargetDisplay (int displayNumber)	Set the radar mini-map to use a particular display. Displays or monitors are numbered from 1 to 8.
SetDisplay (GameObject centre, float range)	Set the world space position of the radar system. Use when you want the radar system to "move" with gameobject.
SetDisplay (Vector3 centre, float range)	Set the world space position of the radar system. Use when you want the radar system to be fixed to a given location or will move infrequently.
SetFactionId (int itemIndex, int factionId)	Set the factionId of the radarItem. NOTE: For performance reasons, it doesn't check if this is non-empty radar item. For ships, the itemIndex is stored as RadarId. e.g. shipControlModule.shipInstance.RadarId. For surfaceTurretModule, the itemIndex is stored as RadarId.
SetItemPosition	Set a new world space position for a radar item. This is useful if you only want to update the position. See also

Property or Method	Description
(int itemIndex, Vector3 wsPosition)	UpdateItem(..). NOTE: For performance reasons, it doesn't check if this is non-empty radar item.
SetItemType(int itemIndex, SSCRadarItem.RadarItemType radarItemType)	Set the type of the radarItem. Typically, this will only be performed once for each object (e.g. a ship or ground installation). NOTE: For performance reasons, it doesn't check if this is non-empty radar item.
SetSquadronId (int itemIndex, int squadronId)	Set the factionId of the radarItem. NOTE: For performance reasons, it doesn't check if this is non-empty radar item. For ships, the itemIndex is stored as RadarId. e.g. shipControlModule.shipInstance.RadarId. For surfaceTurretModule, the itemIndex is stored as RadarId.
SetVisibility (int itemIndex, bool isVisibleToRadar)	Set if this item should appear in Radar queries? NOTE: For performance reasons, it doesn't check if this is non-empty radar item. The itemIndex for a ship is stored on the shpInstance as RadarId.
ShowUI()	If the UI (mini-map) has been configured in the Editor under Visuals, show the radar display on the screen.
SSCRadar.GetOrCreateRadar()	Returns the current SSCRadar instance for this scene. If one does not already exist, a new one is created. If there is an on-screen radar mini-map, it will automatically be hidden. To show it call sscRadar.ShowUI().
sscRadar.GetRadarItem (int itemIndex)	Get an active (assigned) radar item. For ships and locations, the itemIndex is stored as RadarId. e.g. shipControlModule.shipInstance.RadarId
sscRadar.RemoveItem (int itemIndex)	Remove a single item from the radar. Use for items created outside SSC. For ships and Locations, you would typically call shipControlModule.DisableRadar() or sscManager.DisableRadar(...).
UpdateItem(int itemIndex, SSCRadarPacket sscRadarPacket)	Send information about a ship or object to the radar system. For SSC ships you generally won't need to do this as we do it for you. However, for your own items that you want the radar system to track, here is where you need to submit your data after initially calling AddItem(..).

A SSCRadarQuery is a class that holds query options typically passed to the GetRadarResults() method.

Property	Description
centrePosition	This is the world-space position around which the radar query emits.
range	The radar range in metres from the centrePosition
is3DQueryEnabled	Uses 3D distances to determine range when querying the radar data.
querySortOrder	The sort order of the results. None is the fastest option and has the lowest performance impact.
factionId	The faction or alliance the item belongs to. This can be used to identify if an item is friend or foe.

Property	Description
	0 = neutral. -1 means not set (ignore factionId in query and return all factions unless factionsToExclude is set).
factionsToInclude	[Optional] An array of factionIds to include from the query results. Only considered sscRadarQuery.factionId is -1.
factionsToExclude	An array of factionIds to exclude from the query results. For example, if factionId is -1, and factionsToExclude is 0, 2, 4, all objects will be returned <i>except</i> those ships and Locations with a factionId of 0 (neutral), 2 and 4. This would be useful if your player ship had a factionId of 2, you had an alliance with factionId 4 and you wanted to see all enemy ships.
squadronsToInclude	[Optional] An array of squadronIds to include from the query results
squadronsToExclude	[Optional] An array of squadronIds to exclude from the query results

Radar API Call Backs

Callback	Description
CallbackOnDrawBlip callbackOnDrawBlip	<p>The name of the custom method that is called when a blip is to be draw on the radar display. Your method must take 4 parameters - Texture2D, Quaternion, Int, and SSCRadarBlip. Your custom method should "paint" the blip onto the texture by modifying the pixels.</p> <pre>public void YourMethod (Texture2D tex, Quaternion displayRotation, int factionId, SSCRadarBlip sscRadarBlip) { // Your code here }</pre>

Surface Turret Module API Methods

Method	Description
ClearWeaponTarget()	Clears all targeting information for the weapon. This should be called if you do not know if the target is a ship or a gameobject.
DeactivateBeams (SSCManager sscManager)	Deactivate any beams that the weapon is currently firing. This gets automatically called is a DamageReceiver component is attached and there is no health.
FireIfReady()	Fire all cannons on the weapon if they are loaded and ready. This is a single shot action. For continuous firing, call SetAutoFire().
ReinitialiseTurretProjectilesAndEffects()	Reinitialises variables required for projectiles and effects of the surface turret. Call after modifying any projectile or effect data for this surface turret.
SetAutoFire()	Sets the weapon to automatically fire if a target is acquired and the weapon is ready.

Method	Description
SetManualFire()	For manually firing the weapon. After this is set, call FireIfReady() to fire the weapon.
SetWeaponTarget(GameObject target)	The turret will track this gameobject.
SetWeaponTargetShip(ShipControlModule targetShipControlModule)	The turret will track this ship
SetWeaponTargetShipDamageRegion (ShipControlModule targetShipControlModule, DamageRegion damageRegion)	The turret will track this ship's localised damage region
WeaponHasLineOfSight (GameObject target, bool directToTarget = false, bool obstaclesBlockLineOfSight = true, bool anyEnemy = true)	<p>Returns whether a weapon has line of sight to a target. If directToTarget is set to true, will raycast directly from the weapon to the target. If directToTarget is set to false, will raycast in the direction the weapon is facing.</p> <p>This method will return true if the raycast hits:</p> <ul style="list-style-type: none"> a) The target, b) An enemy ship (distinguished by faction ID) - even if it is not the target and anyEnemy is true, c) An object that isn't the target (if obstaclesBlockLineOfSight is set to false), d) Nothing. <p>This method will return false if the raycast hits:</p> <ul style="list-style-type: none"> a) A friendly ship (distinguished by faction ID), b) An object that isn't the target (if obstaclesBlockLineOfSight is set to true). c) An enemy ship that is not the target when anyEnemy is false

Surface Turret Module API Call Backs

Callback	Description
CallbackOnDestroy callbackOnDestroy	<p>The name of the custom method that is called immediately before the turret is destroyed. Your method must take 1 parameter of class SurfaceTurretModule.</p> <p>This should be a lightweight method to avoid performance issues. It could be used to update a score or affect the status of a mission.</p>

Ship Docking Station API Methods

Method	Description
AssignShipToDockingPoint (ShipControlModule shipControlModule, ShipDocking shipDocking, int dockingPointIndex)	Assigns a ship to the docking point with index dockingPointIndex, if that docking point is currently available. You can check whether the docking point is available by calling IsDockingPointAvailable(). If the ship and/or

Method	Description
	ShipDocking components are not initialised, they will be automatically initialised if the docking point is available. See also <code>UnassignDockingPoint(dockingPointIndex)</code> .
<code>AssignShipToDockingPoint (ShipControlModule shipControlModule, ShipDocking shipDocking)</code>	Assigns a ship to the first available docking point (if any are available). If the ship and/or ShipDocking components are not initialised, they will be automatically initialised if a docking point is available. See also <code>UnassignDockingPoint(dockingPointIndex)</code> .
<code>DockShip</code> <code>(int dockingPointIndex)</code>	Start the docking process for an initialised ship at a valid (zero-based) docking point index. The ship must be Undocked. If this is an AI ship and there is an entry path, it will go into the docking state. Otherwise, it will become Docked. The ship must already be assigned to the Docking Point. See also <code>UnDockShip(..)</code> , <code>AssignShipToDockingPoint(..)</code>
<code>DockShip</code> <code>(ShipControlModule shipControlModule)</code>	Start the docking process for an initialised ship at a valid (zero-based) docking point index. The ship must be Undocked. If this is an AI ship and there is an entry path, it will go into the docking state. Otherwise, it will become Docked. The ship must already be assigned to the Docking Point. See also <code>UnDockShip(..)</code> , <code>AssignShipToDockingPoint(..)</code>
<code>GetDockingPoint (int dockingPointIndex)</code>	Safely retrieve a ShipDockingPoint from the ShipDockingStation. Example: <code>ShipDockingPoint shipDockingPoint = shipDockingStation.GetDockingPoint(shipDocking.DockingPointId);</code>
<code>GetDockingPointIndex (int shipId)</code>	Get the zero-based docking point index which a ship is assigned to. If the ship is not initialised or the ship is not assigned to a docking point on this docking station, it will return <code>ShipDockingStation.NoDockingPoint (-1)</code> . Example: <code>int dpIdx = GetDockingPointIndex(shipControlModule.shipInstance.shipId);</code>
<code>GetAssignedShip (int dockingPointIndex)</code>	If a ship is assigned to this docking point, the ShipControlModule for that ship is returned. If the ShipDockingStation is not initialised, or the docking point index is invalid, or there is no assigned ship, this method will return null. NOTE An assigned ship can have a DockingState of Docked, Undocking, or Docking. See also <code>GetDockedShip(..)</code> .
<code>GetDockedShip (int dockingPointIndex)</code>	If the ShipDockingStation is initialised, and the docking point is valid, and a ship is assigned to the docking point, and the ship's DockingStatus is Docked, then return the ShipControlModule. For all other scenarios, return null.
<code>GetDockingPointEntryPathguidHash</code> <code>(int dockingPointIndex)</code>	Get the guidHash of the Entry Path for a docking point using the zero-based index. Use <code>sscManager.GetPath(guidHash)</code> to get the Path. If the guidHash == 0 there is no path assigned, the station is not initialised, or the docking point index is invalid.

Method	Description
GetDockingPointExitPathguidHash (int dockingPointIndex)	Get the guidHash of the Exit Path for a docking point using the zero-based index. Use sscManager.GetPath(guidHash) to get the Path. If the guidHash == 0 there is no path assigned, the station is not initialised, or the docking point index is invalid.
GetStationVelocity()	If the Docking Station is moving (i.e. when it is a mothership), this returns the world space velocity, else it returns Vector3.Zero.
ImportDockingPointDataFromJson (string folderPath, string fileName)	Import a json file from disk and return as list of ShipDockingPoints.
InitialiseDockingPointsPaths (Vector3 positionOffset, Quaternion rotationDelta)	Remember initial Entry/Exit docking path location positions for all ShipDockingPoints. Offset and rotate them as indicated. Initial positions may need to be modified if the ShipDockingStation has moved in the scene from where the Entry/Exit Paths were first setup. Also sets the initial velocity, angular velocity and anchorPoint.
IsDockingPointAvailable (int dockingPointIndex)	Is the docking point with index dockingPointIndex currently available?
IsShipAssigned (int shipId)	Is the ship assigned to any docking points of the Ship Docking Station? The ship does not have to be docked, to be assigned.
IsShipDocked (int shipId)	Is the ship docked at any docking point of this Ship Docking Station?
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.
SaveDockingPointDataAsJson (string filePath)	Save the list of docking points for this ShipDockingStation to a json file on disk.
UnassignDockingPoint (int dockingPointIndex)	Attempt to unassign any ship allocated to a docking point within the ShipDockingStation.
UnDockShip (int dockingPointIndex)	Undock an initialised ship that currently has ShipDocking.DockingState of Docked at a valid (zero-based) docking point index. If the docked ship is an AI ship it will go into the Undocking state (from SSC v1.3.3+ it no longer requires an Exit Path). See also UnDockShip(..), AssignShipToDockingPoint(..). WARNING: Do NOT call this from the onPreUndock event.
UnDockShip (ShipControlModule shipControlModule)	Undock an initialised ship that currently has ShipDocking.DockingState of Docked at a valid (zero-based) docking point index. If the docked ship is an AI ship it will go into the Undocking state (from SSC v1.3.3+ it no longer requires an Exit Path). See also UnDockShip(..), AssignShipToDockingPoint(..) WARNING: Do NOT call this from the onPreUndock event.

Ship Docking API Methods

See also Player Input Module (AI-Assist) API Methods for AI-assisted docking.

Method	Description
<code>GetDockingAdapterPosition (int adapterNumber)</code>	Get the world space position of a docking adapter for the attached ship. Currently only supports adapter number 1.
<code>GetDockingAdapteRotation (int adapterNumber)</code>	Get the world space rotation of a docking adapter for the attached ship. Currently only supports adapter number 1.
<code>GetDockSnapToPosition()</code>	Return the current dockSnapToPos axes.
<code>GetState()</code>	Return the current docking state
<code>GetStateInt()</code>	Return the current docking state as an integer.
<code>RemoveListeners()</code>	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. This is automatically called by <code>OnDestroy</code> .
<code>SetDockSnapToPosition (DockSnapTo newDockSnapTo)</code>	Set which position axes to snap to when a ship gets close to the docking point, and is becomes "Docked". The snap amount can be affected by the Landing Distance Precision.
<code>SetDockSnapToRotation (DockSnapTo newDockSnapTo)</code>	Set which rotation axes to snap to when a ship gets close to the docking point, and is becomes "Docked". The snap amount can be affected by the Landing Angle Precision.
<code>SetState (DockingState dockingState)</code>	Set the docking state of the ship. When undocking, the velocity of <code>ShipDockingStation</code> (mothership) is considered. If configured, a custom method is called after the state has been changed. See Ship Docking API Call Backs below.
<code>StopOnPostDocked()</code>	Prevent a delayed Post Docked event from running once the delay has been triggered by the ship has docked.
<code>StopOnPostDockingHover()</code>	Prevent a delayed Post Docking Hover event from running once the delay has been triggered by the ship reaching the hover point while docking.
<code>StopOnPostDockingStart()</code>	Prevent a delayed Post Docking Start event from running once the delay has been triggered by the ship starting to dock.
<code>StopOnPostUndocked()</code>	Prevent a delayed Post Undocked event from running once the delay has been triggered by the ship finishing the docking manoeuvre.
<code>StopOnPostUndockingHover()</code>	Prevent a delayed Post Undocking Hover event from running once the delay has been triggered by the ship reaching the hover point while undocking.
<code>StopOnPostUndockingStart()</code>	Prevent a delayed Post Undocking Start event from running once the delay has been triggered by the ship starting to undock.

Ship Docking API Call Backs

Callback	Description
CallbackOnStateChange callbackOnStateChange	<p>The name of the custom method that is called immediately after the state is changed. Your method must take 4 parameters: shipDocking (never null), shipControlModule (never null), shipAllInputModule (could be null) and previousDockingState.</p> <p>This should be a lightweight method to avoid performance issues. Your method will NOT be called if ShipDocking.IsInitialised is false.</p> <p>See also onPostDockingHover, onPostUndockingHover events.</p>

Ship Display Module API Properties

These can be found under “Public Variables and Properties” in the ShipDisplayModule.cs script. Comments and descriptions are included.

[IMPORTANT] If you are scripting HUD elements at runtime during the Awake() or Start() events in your game, it is possible that they run before the ShipDisplayModule is initialised even though you have “Initialise On Start” enabled in “General Settings”. If this happens, you can simply write the following code in your game (you just need a reference to shipDisplayModule from the scene):

```
if (!shipDisplayModule.IsInitialised) { shipDisplayModule.Initialise(); }
```

Ship Display Module (General) API Methods

Method	Description
Initialise ()	Initialise the ShipDisplayModule. Either set initialiseOnStart to false and call this method in your code, or set initialiseOnStart to true in the inspector and don't call this method.
IsSourceShip (ShipControlModule shipControlModule) IsSourceShip (Ship ship)	Check to see if the ship is the same as the sourceShip currently assigned to the HUD. If either are null, the method will return false.
ReinitialiseVariables ()	Call this if you modify any of the following at runtime. <ol style="list-style-type: none"> 1) displayRectList 2) The displayReticlePanel size 3) Add or remove the MainCamera tag from a camera
ShowHUD ()	Show the heads-up display
HideHUD ()	Hide the heads-up display
SetCamera (Camera camera)	Set or assign the main camera used by the heads-up display for calculations
SetCanvasTargetDisplay (int displayNumber)	Set the HUD to use a particular display. Displays or monitors are numbered from 1 to 8.
SetHUDDOffset (float offsetX, float offsetY)	Set the offset (position) of the HUD. If the module has been initialised, this will also re-position the HUD.

Method	Description
SetHUDSize (float width, float height)	Set the size of the HUD overlay image and text. If the module has been initialised, this will also resize the HUD. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.
SetPrimaryColour (Color32 newColour) SetPrimaryColour (Color newColour)	Set the primary colour of the heads-up display. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the HUD with the appropriate brightness.
SetBrightness (float newBrightness)	Set the overall brightness of the HUD.
SetCanvasSortOrder(int newSortOrder)	Set the sort order in the scene of the HUD. Higher values appear on top.
ToggleHUD ()	Turn on or off the HUD. Has no effect if not initialised. See also ShowHUD() and HideHUD()

Ship Display Module (Cursor) API Methods

Method	Description
ShowCursor ()	Show the hardware (mouse) cursor. This also restarts the countdown auto-hide timer if that is enabled.
HideCursor ()	Hide the hardware (mouse) cursor. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.
CentreCursor ()	Centre the hardware (mouse) cursor in the centre of the screen. WARNING: This will wait until the next frame before it returns. Best used when a user closes a menu and returns to flying conditions.
ToggleCursor()	Toggle the hardware (mouse) cursor on or off. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.

Ship Display Module (Display Reticule) API Methods

Method	Description
ShowDisplayReticle()	Show the Display Reticle on the HUD. The HUD will automatically be shown if it is not already visible.
HideDisplayReticle()	Hide or turn off the Display Reticle
GetDisplayReticleGuidHash (int index)	Returns the guidHash of the Reticle in the list given the index or zero-based position in the list. Will return 0 if no matching Reticle is found. Will return 0 if the module hasn't been initialised.
GetDisplayReticleGuidHash (string spriteName)	Returns the guidHash of the Reticle in the list given the name of the sprite. Will return 0 if no matching Reticle is found.

Method	Description
	WARNING: This will increase GC. Use <code>GetDisplayReticleGuidHash (int index)</code> where possible.
<code>GetDisplayReticle (int guidHash)</code>	Get a DisplayReticle given its guidHash. See also <code>GetDisplayReticleGuidHash(..)</code> . Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.
<code>GetDisplayReticleSprite (int guidHash)</code>	Get the UI sprite (image) for a DisplayReticle. See also <code>GetDisplayReticleGuidHash(..)</code> .
<code>ChangeDisplayReticle (int guidHash)</code>	Change the DisplayReticle sprite on the UI panel. See also <code>GetDisplayReticleGuidHash(..)</code> .
<code>SetDisplayReticleOffset (Vector2 offset)</code>	Set the offset (position) of the Display Reticle on the HUD. If the module has been initialised, this will also re-position the Display Reticle. Same as <code>SetDisplayReticleOffset(offset.x, offset.y)</code>
<code>SetDisplayReticleOffset (float offsetX, float offsetY)</code>	Set the offset (position) of the Display Reticle on the HUD. If the module has been initialised, this will also re-position the Display Reticle. Input values range from -1 to 1.
<code>SetDisplayReticleColour (Color32 newColour)</code> <code>SetDisplayReticleColour (Color newColour)</code>	Set the active Display Reticle colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the Reticle with the appropriate brightness.
<code>ToggleDisplayReticle()</code>	Show or Hide the Display Reticle on the HUD. The HUD will automatically be shown if required.
<code>ValidateReticleList()</code>	Create a new list if required

Ship Display Module (Altitude and Speed) API Methods

Method	Description
<code>ShowAltitude ()</code>	Attempt to show the Altitude indicator. Turn on HUD if required.
<code>HideAltitude ()</code>	Attempt to turn off the Altitude indicator
<code>SetAltitudeTextColour (Color32 newColour)</code> <code>SetAltitudeTextColour (Color newColour)</code>	Set the altitude text colour on the heads-up display. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the HUD with the appropriate brightness.
<code>ShowAirSpeed ()</code>	Attempt to show the Air Speed indicator. Turn on HUD if required.
<code>HideAirSpeed ()</code>	Attempt to turn off the Air Speed indicator
<code>SetAirSpeedTextColour (Color32 newColour)</code> <code>SetAirSpeedTextColour (Color newColour)</code>	Set the air speed text colour on the heads-up display. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the HUD with the appropriate brightness.

Ship Display Module (Display Attitude) API Methods

Most methods require that the ShipDisplayModule to be initialised. Attitude properties can be set in the editor or via scripts at runtime.

Method	Description
HideAttitude()	Attempt to turn off the Attitude display
SetDisplayAttitudeMaskSprite (Sprite newSprite)	Set the sprite that will mask the Display Attitude scrollable sprite
SetDisplayAttitudePrimaryColour (Color newColour)	Set the Display Attitude primary colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the attitude with the appropriate brightness.
SetDisplayAttitudeScrollSprite (Sprite newSprite)	Set the Display Attitude scroll sprite. E.g., SSCUIPitchLadder1
SetDisplayAttitudeSize (float width, float height)	Set the normalised size of the scrollable Display Attitude. Currently width is always 1.0
SetDisplayAttitudeOffset (float offsetX, float offsetY)	Set the normalised offset (position) of the Display Attitude on the HUD. If the module has been initialised, this will also re-position the Display Attitude.
SetDisplayAttitudeSpriteBorderWidth (float newBorderWidth)	Set the border width of the Scroll Sprite (texture) in pixels. The attitude scroll image should be -90 to 90 deg. Ideally it should have a "border" at the top and bottom of blank (transparent) space to prevent +-90 being right at the top/bottom of the HUD. The image is autoscaled to the full height of the default HUD screen size.
ShowAttitude()	Attempt to show the scrollable Attitude. Turn on HUD if required.

Ship Display Module (Display Fade) API Methods

All methods require that the ShipDisplayModule to be initialised. Fade properties can be set in the editor or via scripts at runtime.

Method	Description
SetDisplayFadeColour (Color newFadeColour)	Set the colour the display will fade in from or out to.
StartDisplayFade (bool isFadeIn)	Start fading in or out the display.
StopDisplayFade()	If fading in or out, turn the fade off.

Ship Display Module (Display Flicker) API Methods

All methods require that the ShipDisplayModule to be initialised. Flicker properties can be set in the editor or via scripts at runtime.

Method	Description
FlickerOn (float duration)	Flicker the HUD on/off. Override the Flicker Default Duration. Show the HUD when flickering finishes.
FlickerOff (float duration)	Flicker the HUD on/off. Override the Flicker Default Duration. Hide the HUD when flickering finishes.

Ship Display Module (Display Gauge) API Methods

Where possible, always use the methods that take the DisplayGauge or guidHash of the gauge as a parameter.

Most methods require that the ShipDisplayModule to be initialised.

Method	Description
AddGauge (string gaugeName, string gaugeText)	Add a new gauge to the HUD. By design, they are not visible at runtime when first added.
AddGauge (DisplayGauge displayGauge)	Add a gauge to the HUD using a displayGauge instance. Typically, this is used with CopyDisplayGauge(..).
DeleteGauge (int guidHash)	Delete a gauge from the HUD. NOTE: It is much cheaper to HideDisplayGauge(..) than completely remove it.
CopyDisplayGauge (DisplayGauge displayGauge, string NameOfCopy)	Create a copy of an existing DisplayGauge, and give it a new name. Call AddGauge(newDisplayGauge) to make it useable in the game.
GetDisplayGaugeGuidHash (int index)	Returns the guidHash of the Gauge in the list given the index or zero-based position in the list. Will return 0 if no matching Gauge is found.
GetDisplayGaugeIndex (int guidHash)	Get the zero-based index of the Gauge in the list. Will return -1 if not found.
GetDisplayGauge (int guidHash)	Get a DisplayGauge given its guidHash. Will return null if guidHash parameter is 0, it cannot be found. See also GetDisplayGaugeGuidHash(..)
GetDisplayGauge (string displayGaugeName)	Get the display gauge give the description title of the gauge. WARNING: This will increase Garbage Collection (GC). Where possible use GetDisplayGauge(guidHash) and/or GetDisplayGaugeGuidHash(index)
HideDisplayGauge (int guidHash)	Hide or turn off the Display Gauge
HideDisplayGauge (DisplayGauge displayGauge)	Hide or turn off the Display Gauge
HideDisplayGauges()	Hide or turn off all Display Gauges. ShipDisplayModule must be initialised.
ImportGaugeFromJson (string folderPath, string fileName)	Import a json file from disk and return as DisplayGauge
RefreshGaugesSortOrder ()	After adding or moving DisplayGauges, they may need to be sorted to have the correct z-order in on the HUD.
SaveGaugeAsJson (DisplayGauge displayGauge, string filePath)	Save the Gauge to a json file on disk.
SetDisplayGaugeValueAffectsColourOn (DisplayGauge displayGauge, Color lowColour, Color mediumColour, Color highColour)	The foreground colour of the gauge will be determined by the gauge value and the low, medium and high colours. LowColour = value of 0, MediumColour when value is Medium Colour Value, and HighColour when value is 1.0
SetDisplayGaugeValueAffectsColourOff (DisplayGauge displayGauge, Color newForegroundColour)	The value of the gauge does not affect the foreground colour. When turning off this feature the new foreground colour would typically be the old foregroundHighColour.

Method	Description
SetDisplayGaugeOffset (DisplayGauge displayGauge, float offsetX, float offsetY)	Set the offset (position) of the Display Gauge on the HUD. If the module has been initialised, this will also re-position the Display Gauge.
SetDisplayGaugeMediumColourValue (DisplayGauge displayGauge, float newValue)	Set the gauge value at which the foreground medium colour should be set. The default value is 0.5.
SetDisplayGaugeSize (DisplayGauge displayGauge, float width, float height)	Set the size of the Gauge Panel. If the module has been initialised, this will also resize the Gauge Panel. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.
SetDisplayGaugeValue (DisplayGauge displayGauge, float gaugeValue)	Update the value or reading of the gauge. If Value Affects Colour (isColourAffectByValue) is enabled, the foreground colour of the gauge will also be updated. Values should be in range 0.0 to 1.0.
SetDisplayGaugeLabel (DisplayGauge displayGauge, string gaugeLabel)	Update the text of the gauge label. This only applies to numeric gauges with labels. For non-numeric gauges see SetDisplayGaugeText(..).
SetDisplayGaugeLabelAlignment (DisplayGauge displayGauge, TextAnchor textAlignment)	Update the position of the label within the gauge panel. Only applies to numeric gauges with a label. See also SetDisplayGaugeTextAlignment(..).
SetDisplayGaugeText (DisplayGauge displayGauge, string gaugeText)	Update the text of the gauge. For numeric gauges with a label see SetDisplayGaugeLabel(..).
SetDisplayGaugeTextAlignment (DisplayGauge displayGauge, TextAnchor textAlignment)	Update the position of the text within the gauge panel
SetDisplayGaugeTextFont (DisplayGauge displayGauge, Font font)	Set the font of the DisplayGauge Text component
SetDisplayGaugeTextFontSize (DisplayGauge displayGauge, bool isBestFit, int minSize, int maxSize)	Set the font size of the display gauge text. If isBestFit is false, maxSize is the font size set.
SetDisplayGaugeTextColour (DisplayGauge displayGauge, Color newColour)	Set the Display Gauge text colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge text with the appropriate brightness.
SetDisplayGaugeTextDirection (DisplayGauge displayGauge, DisplayGauge.DGTextDirection textDirection)	Update the rotation of the text within the gauge panel
SetDisplayGaugeTextFontStyle (DisplayGauge displayGauge, FontStyle fontStyle)	Set the font style of the DisplayGauge Text component
SetDisplayGaugeType (DisplayGauge displayGauge, DisplayGauge.DGType dgType)	Set the type or style of the gauge.
SetDisplayGaugeForegroundColour (DisplayGauge displayGauge, Color newColour)	Set the Display Gauge foreground colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge foreground with the appropriate brightness.

Method	Description
SetDisplayGaugeForegroundSprite (DisplayGauge displayGauge, Sprite newSprite)	Set the Display Gauge foreground sprite. This is used to render the gauge value by partially filling it.
SetDisplayGaugeBackgroundColour (DisplayGauge displayGauge, Color newColour)	Set the Display Gauge background colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge background with the appropriate brightness.
SetDisplayGaugeBackgroundSprite (DisplayGauge displayGauge, Sprite newSprite)	Set the Display Gauge background sprite. This is used to render the background image of the gauge.
SetDisplayGaugeFillMethod (DisplayGauge displayGauge, DisplayGauge.DGFillMethod fillMethod)	Set the Display Gauge fill method. This determines how the gauge is filled
SetDisplayGaugeKeepAspectRatio (DisplayGauge displayGauge, bool isKeepAspectRatio)	Sets whether or not the foreground and background sprites keep their original texture aspect ratio. This can be useful when creating circular gauges.
ShowDisplayGauge(int guidHash)	Show the Display Gauge on the HUD. The HUD will automatically be shown if it is not already visible.
ShowDisplayGauge (DisplayGauge displayGauge)	Show the Display Gauge on the HUD. The HUD will automatically be shown if it is not already visible.
ShowDisplayGauges()	Show or turn on all Display Gauges. ShipDisplayModule must be initialised. The HUD will automatically be shown if it is not already visible.
ValidateGaugeList()	Create a new list if required

Ship Display Module (Display Heading) API Methods

Most methods require that the ShipDisplayModule to be initialised. Heading properties can be set in the editor or via scripts at runtime.

Method	Description
SetDisplayHeadingIndicatorSprite (Sprite newSprite)	Set the small sprite that will indicate or point to the heading on the HUD
SetDisplayHeadingMaskSprite (Sprite newSprite)	Set the sprite that will mask the Display Heading scrollable sprite
SetDisplayHeadingIndicatorColour (Color newColour)	Set the Display Heading small indicator colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the heading with the appropriate brightness.
SetDisplayHeadingPrimaryColour (Color newColour)	Set the Display Heading primary colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the heading with the appropriate brightness.
SetDisplayHeadingScrollSprite (Sprite newSprite)	Set the Display Heading scroll sprite.

Method	Description
SetDisplayHeadingSize (float width, float height)	Set the normalised size of the scrollable Display Heading. Currently the height is always 1.0
SetDisplayHeadingOffset (float offsetX, float offsetY)	Set the normalised offset (position) of the Display Heading on the HUD. If the module has been initialised, this will also re-position the Display Heading. The Horizontal and Vertical offset is from the centre. Range between -1 and 1.
HideHeading()	Attempt to turn off the scrollable Heading
HideHeadingIndicator()	Attempt to turn off the Heading indicator
ShowHeading()	Attempt to show the scrollable Heading. Turn on HUD if required.
ShowHeadingIndicator()	Attempt to turn on the Heading indicator

Ship Display Module (Display Message) API Methods

Where possible, always use the methods that take the DisplayMessage or guidHash of the message as a parameter. These are much more efficient than using the name of the message which will incur GC overhead.

Most methods require that the ShipDisplayModule to be initialised.

Method	Description
ShowDisplayMessage (int guidHash) ShowDisplayMessage (DisplayMessage displayMessage)	Show the Display Message on the HUD. The HUD will automatically be shown if it is not already visible. If fade-in duration is greater than 0 seconds, it will fade in over that period of time.
ShowDisplayMessageInstant (int guidHash) ShowDisplayMessageInstant (DisplayMessage displayMessage)	Show the Display Message on the HUD instantly by ignoring fade settings. The HUD will automatically be shown if it is not already visible.
ShowDisplayMessages ()	Show or turn on all Display Messages. ShipDisplayModule must be initialised. The HUD will automatically be shown if it is not already visible.
HideDisplayMessage (int guidHash) HideDisplayMessage (DisplayMessage displayMessage)	Hide or turn off the Display Message. If fade-out duration is greater than 0 seconds, the message will fade out over that period of time.
HideDisplayMessageInstant (int guidHash) HideDisplayMessageInstant (DisplayMessage displayMessage)	Hide or turn off the Display Message
HideDisplayMessages ()	Hide or turn off all Display Messages. ShipDisplayModule must be initialised.
AddMessage (string messageName, string messageText)	Add a new message to the HUD and returns a reference to the Display Message. By design, they are not visible at runtime when first added.
AddMessage (DisplayMessage displayMessage)	Add a message to the HUD using a displayMessage instance. Typically, this is used with CopyDisplayMessage(..).

Method	Description
DeleteMessage (int guidHash)	Delete a message from the HUD. NOTE: It is much cheaper to HideDisplayMessage (..) than completely remove it.
CopyDisplayMessage (DisplayMessage displayMessage, string NameOfCopy)	Create a copy of an existing DisplayMessage, and give it a new name. Call AddMessage(newDisplayMessage) to make it useable in the game.
GetDisplayMessageGuidHash (int index)	Returns the guidHash of the Message in the list given the index or zero-based position in the list. Will return 0 if no matching Message is found. Will return 0 if the module hasn't been initialised.
GetDisplayMessage (int guidHash)	Get a DisplayMessage given its guidHash. See also GetDisplayMessageGuidHash(..). Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.
GetDisplayMessage (string displayName)	Get the display message give the description title of the message. WARNING: This will increase Garbage Collection (GC). Where possible use GetDisplayMessage(guidHash) and/or GetDisplayMessageGuidHash(index).
GetDisplayMessageIndex (int guidHash)	Get the zero-based index of the Message in the list. Will return -1 if not found.
ImportMessageFromJson (string folderPath, string fileName)	Import a json file from disk and return as DisplayMessage.
IsDisplayMessageShown (DisplayMessage displayMessage)	Is the Display Message currently being shown on the HUD?
ShowDisplayMessageBackground (DisplayMessage displayMessage)	Show the Display Message background on the HUD. The display the actual message, you would need to call ShowDisplayMessage(..).
HideDisplayMessageBackground (DisplayMessage displayMessage)	Hide the Display Message background on the HUD. The hide the actual message, you would need to call HideDisplayMessage(..).
SaveMessageAsJson (DisplayMessage displayMessage, string filePath)	Save the Message to a json file on disk.
SetDisplayMessageOffset (DisplayMessage displayMessage, float offsetX, float offsetY)	Set the offset (position) of the Display Message on the HUD. If the module has been initialised, this will also re-position the Display Message. Parameter: offsetX is horizontal offset from centre. Range between -1 and 1 Parameter: offsetY is vertical offset from centre. Range between -1 and 1
SetDisplayMessageSize (DisplayMessage displayMessage, float width, float height)	Set the size of the Message Panel. If the module has been initialised, this will also resize the Message Panel. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.

Method	Description
SetDisplayMessageBackgroundColour (DisplayMessage displayMessage, Color newColour)	Set the Display Message background colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the message background with the appropriate brightness.
SetDisplayMessageScrollDirection (DisplayMessage displayMessage, int scrollDirection)	Set the Display Message scroll direction. USAGE: SetDisplayMessageScrollDirection(displayMessage, DisplayMessage.ScrollDirectionLR)
SetDisplayMessageScrollFullscreen (DisplayMessage displayMessage, bool isScrollFullscreen)	Set the Display Message to scroll across or up/down the full screen regardless of the message width and height. Can also be set directly with displayMessage.isScrollFullscreen = true;
SetDisplayMessageScrollSpeed (DisplayMessage displayMessage, float scrollSpeed)	Set the Display Message scroll speed. Can also be set directly with: displayMessage.scrollSpeed = scrollSpeed;
SetDisplayMessageText (DisplayMessage displayMessage, string messageText)	Update the text of the message.
SetDisplayMessageTextAlignment (DisplayMessage displayMessage, TextAnchor textAlignment)	Update the position of the text within the message panel
SetDisplayMessageTextFont (DisplayMessage displayMessage, Font font)	Set the font of the DisplayMessage Text component. The default is Arial, but you can supply your own.
SetDisplayMessageTextFontSize (DisplayMessage displayMessage, bool isBestFit, int minSize, int maxSize)	Set the font size of the display message text. If isBestFit is false, maxSize is the font size set.
SetDisplayMessageTextColour (DisplayMessage displayMessage, Color newColour)	Set the Display Message text colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the message with the appropriate brightness.

Ship Display Module (Display Target) API Methods

Where possible, always use the methods that take the DisplayTarget or guidHash of the target as a parameter.

Most methods require that the ShipDisplayModule to be initialised.

Method	Description
AddTarget (int guidHashDisplayReticle)	Add a new Target to the HUD and returns a reference to the Display Target. By design, they are not visible at runtime when first added.
AddTargetSlots (DisplayTarget displayTarget, int numberToAdd)	Add another DisplayTarget slot to a DisplayTarget. This allows you to display another copy of the target on the HUD. If the DisplayTarget has not been initialised, the new slot panel will be added to the scene but this method will return null. This automatically updates displayTarget.maxNumberOfTargets.
AssignDisplayTargetSlot (int displayTargetIndex, int slotIndex, int radarItemIndex, uint radarItemSequenceNumber, bool isAutoShow)	Assign a radar target to a DisplayTarget's slot. If isAutoShow is true, the TargetDisplay will be shown if there is a valid radarItem, else it will be hidden.
DeleteTarget (int guidHash)	Delete a Target from the HUD. NOTE: It is much cheaper to HideDisplayTarget (..) than completely remove it.

Method	Description
DeleteTargetSlots (int guidHash, int numberToDelete)	Delete or remove a DisplayTarget slot from the HUD. This is an expensive operation. It is much cheaper to HideDisplayTargetSlot(..) than completely remove it. Automatically updates displayTarget.maxNumberOfTargets. NOTE: You cannot remove slot 0.
GetAssignedDisplayTarget (int displayTargetIndex, int slotIndex)	Get the radarItemKey for a DisplayTarget's slot. SSCRadarItemKey.radarItemIndex will be -1 if there is no target.
GetAssignedDisplayTargetRadarItem (DisplayTargetSlot displayTargetSlot)	Get the SSCRadarItem currently assigned to a DisplayTarget slot (instance).
GetDisplayTarget (int guidHash)	Get a DisplayTarget given its guidHash. See also GetDisplayTargetGuidHash(..). Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.
GetDisplayTargetByIndex (int index)	Get a DisplayTarget given a zero-based index in the list.
GetDisplayTargetGuidHash (int index)	Returns the guidHash of the Target in the list given the index or zero-based position in the list. Will return 0 if no matching Target is found. Will return 0 if the module hasn't been initialised.
GetDisplayTargetIndex (int guidHash)	Get the zero-based index of the Target in the list. Will return -1 if not found.
GetDisplayTargetName (DisplayTarget displayTarget) (int index)	Get the (sprite) name of the DisplayTarget using either the DisplayTarget instance or the zero-based index in the list of DisplayTargets on the HUD. WARNING: This will create GC, so not recommended to be called each frame. This is typically used for debugging purposes only.
GetDisplayTargetSlotByIndex (DisplayTarget displayTarget, int slotIndex)	Get a DisplayTargetSlot given the zero-based slot index. This is an instance of a DisplayTarget.
HideDisplayTarget(int guidHash) HideDisplayTarget (DisplayTarget displayTarget)	Hide or turn off all slots of the Display Target
HideDisplayTargets ()	Hide or turn off all slots of all Display Targets.
HideDisplayTargetSlot (DisplayTargetSlot displayTargetSlot)	Hide the Display Target slot on the HUD. By design, if the HUD is not shown, the Target in this slot will not be show.
SetDisplayTargetOffset (DisplayTarget displayTarget, int slotIndex, float offsetX, float offsetY)	Set the offset (position) of the Display Target slot on the HUD. If the module has been initialised, this will also re-position the Display Target. Horizontal offset from centre. Range between -1 and 1. Vertical offset from centre. Range between -1 and 1.
SetDisplayTargetOffset (DisplayTargetSlot displayTargetSlot, float offsetX, float offsetY)	Set the offset (position) of the Display Target slot on the HUD. If the module has been initialised, this will also re-position the Display Target.
SetDisplayTargetPosition (DisplayTargetSlot displayTargetSlot, float offsetX, float offsetY)	Move the DisplayTarget slot to the correct 2D position on the HUD, based on a 3D world space position. If the camera has

Method	Description
	not been automatically or manually assigned, the DisplayTarget will not be moved.
SetDisplayTargetReticle (int guidHash, int guidHashDisplayReticle)	Set or change the Reticle assigned to a DisplayTarget. The Reticle must belong to the list of available reticles for the HUD.
SetDisplayTargetReticleColour (DisplayTarget displayTarget, Color newColour)	Set the Display Target Reticle colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the reticle with the appropriate brightness.
SetTargetsViewportOffset (float offsetX, float offsetY)	Sets the viewport offset from the centre of the screen. Values can be between -1.0 and 1.0 with 0,0 depicting the centre of the screen. This is the area of the screen in which Targets will be visible. See also SetTargetsViewportSize(..).
SetTargetsViewportSize (float width, float height)	Sets the clipped viewable area of the screen that DisplayTargets can be shown. When Targets are outside this area, they will be hidden. Width and height values are between 0.1 and 1.0 of the total screen width or height. See also SetTargetsViewportOffset(..).
ShowDisplayTarget (int guidHash) ShowDisplayTarget (DisplayTarget displayTarget)	Show the Display Target on the HUD. By design, if the HUD is not shown, the Targets will not be show.
ShowDisplayTargets ()	Show or turn on all Display Targets.
ShowDisplayTargetSlot (DisplayTargetSlot displayTargetSlot)	Show the Display Target slot on the HUD. By design, if the HUD is not shown, the Target in this slot will not be show.

Ship Display Module API Call Backs

Callback	Description
CallbackOnBrightnessChange callbackOnBrightnessChange	<p>The name of the custom method that is called immediately after brightness has changed. Your method must take 1 float parameter. This should be a lightweight method to avoid performance issues. It could be used to update your custom HUD elements.</p> <pre>shipDisplayModule.callbackOnBrightnessChange = YourMethod;</pre> <pre>public void YourMethod (float value) { // Adjust the brightness of your elements here }</pre>
CallbackOnSizeChange callbackOnSizeChange	<p>The name of the custom method that is called immediately after the HUD size has changed. This method must take 2 float parameters. It should be a lightweight method to avoid</p>

Callback	Description
	performance issues. It could be used to update your custom HUD elements.

Ship Warp Module (General) API Properties

Property	Description
EnvironmentAmbientSource	Get or set the source of the ambient light. Colour, Gradient or Skybox.
IsInitialised	Is the module initialised?
IsWarpEngaged	Is warp currently engaged?
MaxWarpDuration	Get or set the time, in seconds, that warp will automatically disengage. If it is set to 0, it will not automatically disengage.

Ship Warp Module (Camera) API Properties

Property	Description
CurrentCameraSettingsIndex	The zero-based index of the currently used ShipCameraSettings. Will return - 1 if the module is not initialised, or there are no active camera settings applied.
IsApplyCameraSettingsOnEngage	If there are optional camera settings configured, apply the first one when warp is engaged.

Ship Warp Module (General) API Methods

Method	Description
virtual void DisengageWarp()	Attempt to disengage warp fx for the ship.
virtual void EngageWarp()	Attempt to engage warp fx for the ship
virtual void Initialise()	Initialise the ShipWarpModule. Either set initialiseOnStart to false and call this method in your code, or set initialiseOnStart to true in the inspector and don't call this method.
virtual void RefreshManager()	Update or create pooled sound effects with SSCManager in the scene.
SetEnvironmentAmbientSource (int envAmbientSourceInt)	Attempt to set the environment ambient source (0: Colour, 1: Gradient, 2: Skybox).
SetEnvironmentAmbientSource (EnvAmbientSource newEnvAmbientSource)	Set the environment ambient source (Colour, Gradient or Skybox).
virtual void StopShipMovement()	Attempt to stop the ship from moving.
ToggleWarp()	Attempt to toggle warp on and off.

Ship Warp Module (Ship) API Methods

Method	Description
GetDefaultPitchCurve()	Return the default pitch animation curve.
GetDefaultRollCurve()	Return the default roll animation curve.
SetMaxShakeInterval (float newMaxShakeInterval)	Attempt to set the maxShakeInterval.
SetMinShakeInterval (float newMinShakeInterval)	Attempt to set the minShakeInterval.
SetMaxShakeStrength (float newMaxShakeStrength)	Attempt to set the maxShakeStrength.
SetMaxShakeDuration (float newMaxShakeDuration)	Attempt to set the maxShakeDuration.
SetShipControlModule (ShipControlModule newShipControlModule)	Set the ship being used with the Ship Warp Module.
SetShipPitchCurve (AnimationCurve newAnimationCurve)	Attempt to set the shipPitchCurve
SetMaxShipPitchDown (float newMaxShipPitchDown)	Attempt to set the maxShipPitchDown angle
SetMaxShipPitchDuration (float newMaxShipPitchDuration)	Attempt to set the maximum time, in seconds, the ship will take to pitch up and down.
SetMaxShipPitchUp (float newMaxShipPitchUp)	Attempt to set the maxShipPitchUp angle.
SetMaxShipRollAngle (float newMaxShipRollAngle)	Attempt to set the maxShipRollAngle.
SetMaxShipRollDuration (float newMaxShipRollDuration)	Attempt to set the maxShipDuration.
SetShipRollCurve (AnimationCurve newAnimationCurve)	Attempt to set the shipRollCurve.

Ship Warp Module (Camera) API Methods

Method	Description
CycleCameraSettings()	Attempt to cycle through a list of camera settings. It will skip over any empty settings slots and will not apply the same settings twice in a row.
GetShipCameraModule()	Get the current ship camera module (if any).
SetShipCameraModule (ShipCameraModule newShipCameraModule)	Attempt to set the ship camera module used by the Warp FX

Ship Warp Module (FX) API Methods

Method	Description
virtual void DisableParticleFX()	Attempt to disable the particle systems.
PauseSoundFX()	Attempt to pause the sound FX feature.
SetInnerParticleSystem (ParticleSystem newParticleSystem)	Attempt to update the InnerParticleSystem with another ParticleSystem.

Method	Description
SetMaxSoundInterval (float newMaxSoundInterval)	Attempt to set the maxSoundInterval
SetOuterParticleSystem (ParticleSystem newParticleSystem)	Attempt to update the OuterParticleSystem with another ParticleSystem.
SetSoundFXSet (SSCSoundFXSet newSoundFXSet)	Attempt to update the Sound FX Set.
UnpauseSoundFX()	Attempt to un-pause the sound FX feature.

Ship Warp Module (Events) API Methods

Method	Description
virtual void RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.

Support

For any other questions you have (or any suggestions on how we can improve Sci-Fi Ship Controller), feel free to contact us via our Unity forum (<https://forum.unity.com/threads/594448/>) or on our Discord channel (<https://discord.gg/CjzCK4b>).

Version History

Initial Release: 22 August 2019

Version 1.0.2 – 16 October 2019

[NEW] PlayerInputModule - support for (new) Unity Input System in U2019.1+
 [FIXED] Weapons - NullReferenceException when (I)nserting a duplicate weapon
 [IMPROVED] Sound - Use OGG files rather than MP3

Version 1.0.3 – 21 October 2019

[NEW] Ground Match Smoothing option when Stick to Surface is enabled
 [NEW] Ground Normal History option when Stick to Surface is enabled

Version 1.0.4 – 28 November 2019

[NEW] Location and Path editor in SSC Manager
 [NEW] Location and Path API

Version 1.0.5 – 03 December 2019

[FIXED] When Limit pitch and roll is enabled, pitch is incorrectly set
 [IMPROVED] Player Input Module - Unity Input System can set Pitch/Yaw to use Mouse

Version 1.1.0 – 04 January 2020

[NEW] Ship AI system with API
 [NEW] Asteroids demo using Ship AI
 [FIXED] Multiple Path Locations / Control points are selected without SHIFT key held

- [FIXED] DOTS support should not be enabled before Hybrid Renderer is installed
- [FIXED] PlayerInput - Pitch or Yaw mouse input may not be reset when adding events
- [IMPROVED] Celestials can be placed on a non-default Unity Layer
- [IMPROVED] Celestials can optionally appear below the horizon
- [IMPROVED] City Demo scene includes Ship AI system

Version 1.1.1 – 30 January 2020

- [NEW] Projectiles with DOTS enabled, can hit Entities with a PhysicsCollider
- [NEW] Ship AI API - SSCMath.FindClosestPointOnPath when previous point on path is unknown
- [NEW] Ship API - EnableShip, DisableShip, ResetShip, and AddHealth methods
- [NEW] Camera API - EnableCamera and DisableCamera methods
- [NEW] Rumble or force feedback to controllers with Unity (new) Input System or Rewired.
- [NEW] Take action if a ship becomes stuck (respawn on a path, use respawning mode or call your method).
- [NEW] Respawn Mode - added Respawn On Path option
- [NEW] PlayerInputModule - per axis sensitivity options
- [IMPROVED] ShipAllInputModule - Ship Speed in km/h available in Debug mode in the editor
- [IMPROVED] PlayerInputModule - Ship Speed in km/h available in Debug mode in the editor
- [IMPROVED] SampleShowShipHealth - disable raycast target on Text objects
- [IMPROVED] Faction Id and Squadron Id are now configurable from the Ship Control Module editor

Version 1.1.2 – 13 February 2020

- [NEW] Ship API - SetWeaponFireDirection method added to provide easier changes at runtime
- [NEW] Auto-fire turrets can optionally check line-of-sight
- [NEW] Locations can have a Faction Id
- [NEW] Radar API (in technical preview)
- [NEW] Radar on-screen mini-map (in technical preview)
- [FIXED] PlayerInputModule - Add Events can lose connection to Input Action asset
- [FIXED] Ship AI - maxSpeed is not honoured correctly
- [FIXED] Ship AI - ship's position may become unstable after EnableShip() and velocity is reset/changed.
- [FIXED] Projectiles - when pooling is enabled, projectiles may get prematurely de-spawned
- [FIXED] DemoFlyToLocation does not check for DemoFlyToLocationShipData component
- [IMPROVED] ShipControlModule CallbackOnHit now provides hit and projectile information

Version 1.1.3 – 03 March 2020

- [NEW] Projectile Damage types
- [NEW] Damage Receiver for custom non-ship objects
- [NEW] PlayerInputModule - Auto Cruise option
- [NEW] PlayerInputModule API - EnableInput(), DisableInput(), and ResetInput() methods
- [NEW] Surface Turret Module - add free standing weapons to your scene
- [NEW] Guided Projectiles - turrets can fire projectiles that lock on to their targets
- [NEW] Stick to Ground Surface - Look ahead option to help detect obstacles ahead of the ship
- [NEW] Auto Targeting Module - Use radar to set ship turret or surface turret targets
- [FIXED] Radar - IndexOutOfRangeException: Array index is out of range on screen resize
- [FIXED] Localised Damage Region - ArgumentOutOfRangeException on destroy when no effects object
- [FIXED] Localised Damage Region may not be hit by a projectile if the collider is same size as the region
- [IMPROVED] (F)ind button to select a thruster, wing, control surface, damage region or weapon in scene
- [IMPROVED] Ship turret weapons - Relative Position is set when the Pivot Y transform is set.
- [IMPROVED] Radar API - factionsToExclude query option for faster friend or foe elimination

Version 1.1.4 – 20 April 2020

[NEW] Surface Turret Module supports effects objects
 [NEW] Turrets have optional inaccuracy
 [NEW] Paths - Snap To Mesh option in SSCManager Editor and runtime API
 [NEW] Locations – Snap to Mesh option in SSCManager (runtime API only)
 [NEW] Radar - query options to include/exclude ship squadrons
 [NEW] Call your custom method when a ship is respawned
 [NEW] Ship API - EnableShipMovement and DisableShipMovement methods
 [NEW] Ship Control Module - Respawn can be paused or resumed via API
 [NEW] Projectiles - guided projectile prefab
 [NEW] Ship Docking Station system [Technical Preview]
 [NEW] AI States: Docking and Strafing Run [Technical Preview]
 [NEW] Docking AI Input Behaviour [Technical Preview]
 [NEW] Added current state stage index for AI, to keep track of what has been completed in an AI state
 [FIXED] Surface Turret Module - NullReferenceException before Turret Pivot X added
 [FIXED] Surface Turret Module - ships targeted by radar do not consider their velocity
 [FIXED] Ship Control Module - PID controller regression
 [FIXED] Ship Control Module - damage regions in the editor are not rotated correctly
 [FIXED] Radar - Locations Visible to Radar may not appear on mini-map when scene starts
 [FIXED] PlayerInputModule - Oculus VR secondary fire button held down not set in editor
 [FIXED] Ship AI Input Module - renamed Pitch Yaw Bias to Roll Bias
 [FIXED] Projectiles - Effects may occasionally not work when colliding with an object
 [IMPROVED] Surface Turret Module - automatically receive damage when a Damage Receiver is attached
 [IMPROVED] Turret precision for ship targets
 [IMPROVED] Turrets automatically stop targeting a ship that has been destroyed
 [IMPROVED] Weapons that have no health, cannot aim, reload or fire.
 [IMPROVED] Player Input Module - Initialise on Awake option
 [IMPROVED] Ship collision damage is now automatically scaled based on the mass
 [IMPROVED] Radar - Ships and Locations can have individual blip sizes
 [IMPROVED] Manual - greatly expanded Runtime and API documentation
 [IMPROVED] Full 3D Flight can now match an up direction passed by an AI Behaviour Output

Version 1.1.5 – 28 April 2020

[NEW] ShipControlModule - Thrusters have optional throttle up/down time.
 [NEW] ShipControlModule - Brake Flight Assist
 [NEW] Radar Basics Tutorial
 [NEW] Tutorials link from ShipControlModule editor
 [NEW] 2 new thruster effects and 1 new thruster sound
 [FIXED] ShipAI Follow Path - Argument is out of range when Path has less than 2 assigned Locations
 [FIXED] Radar - ships may stay on the radar after being destroyed
 [IMPROVED] ShipControlModule - (F)ind button will enable gizmo
 [IMPROVED] ShipControlModule - search filter for thrusters in the editor

Version 1.1.6 – 22 June 2020

[NEW] 2 new thruster effects
 [NEW] Thrusters - Effects can be enabled based upon ship velocity
 [NEW] Thrusters - Effects can be enabled or disabled via API for different performance levels
 [NEW] Ship and Surface Turrets - in editor visuals for turning and firing arc
 [NEW] SurfaceTurretModule - callback for when turret is destroyed in API
 [NEW] ProjectModule - Muzzle FX Object option (particle system and/or sound)
 [NEW] Player Input Module - ability to override an axis in code (sample script included)
 [NEW] Ship Display Module (HUD) with API in technical preview
 [FIXED] ShipDockingStation - scene is not updated when all docking point gizmos are toggled

[FIXED] SSCManager - NullReferenceException when adding new Path or Location
 [IMPROVED] Path gizmos are enabled when Activate for editing is clicked in the editor
 [IMPROVED] Effects on included Ship prefabs have less performance overhead
 [IMPROVED] Ship AI Input Module - debugger can display velocity in km/h and m/s
 [IMPROVED] Celestials - demo script supports a second camera
 [IMPROVED] Ship Camera Module - suggest using Target Rotation if Aim To Target is selected
 [IMPROVED] HideCursor demo script is now New Input System aware
 [IMPROVED] Added Ship API methods to the manual

Version 1.1.7 – 8 July 2020

[NEW] Ship Docking - editor runtime debugging option
 [FIXED] Ship Display Module - calling Initialise twice may result in incorrect HUD layout
 [FIXED] SSCManager.Initialise - could not get entities World
 [FIXED] SSCManager - cannot delete locations or paths in U2019.4+
 [FIXED] ShipDockingStation - NullReferenceException when adding docking point and DOTS is enabled
 [FIXED] ShipDocking - adapter should not remain selected when Gizmo is turned off
 [FIXED] ShipDocking - ship may not dock with a ship docking station
 [FIXED] ShipDocking - thruster effects do not turn off when docked
 [FIXED] ShipControlModule - cannot delete wing, control surface, damage region or weapon in U2019.4+
 [FIXED] ShipDisplayModule - cannot delete reticles in U2019.4+
 [IMPROVED] Ship Docking Station - unselected docking points are slightly transparent
 [IMPROVED] Ship Docking Station - new docking points are automatically selected in the scene
 [IMPROVED] Ship Docking Station - docking and undocking
 [IMPROVED] Ship AI Input Module - roll bias

Version 1.1.8 – 19 August 2020

[NEW] PlayerInputModule - Mouse deadzone
 [NEW] ShipDisplayModule - IsSourceShip, CentreCursor API methods
 [NEW] ShipDisplayModule - scrollable messages with API
 [NEW] ShipDisplayModule - on-screen targets with API
 [NEW] ProjectileModule - editor runtime debugging option
 [NEW] ProjectileModule - guided projectiles have adjustable turning speed
 [NEW] HUD - SampleHUDTargets script for on-screen target tracking
 [NEW] Radar API methods to check if blips are in viewable screen viewports
 [NEW] AutoTargetingModule - optional on-screen target tracking
 [NEW] AutoTargetingModule - support for fixed weapons with guided projectiles
 [NEW] Ship AI Input Module - configurable banking and pitch angles
 [FIXED] ShipDisplayModule - HUD does not adjust for screen resizing
 [FIXED] Editors - cannot delete or move list items in U2019.3+
 [FIXED] Radar - GetRadarResults API method ignored query sort order and created GC
 [FIXED] Weapons - turning off Multiple Fire Positions may result in incorrect fire position
 [FIXED] ShipControlModule - NullReferenceException when clicking Visible to Radar at runtime
 [IMPROVED] Ship AI Input Module - runtime editor debugging
 [IMPROVED] Ship Docking is out of technical preview
 [IMPROVED] PlayerInputModule – (new) Unity Input System out of technical preview
 [IMPROVED] PlayerInputModule - detect when legacy Unity Input System is disabled in 2019.2+
 [IMPROVED] Turrets inaccuracy option is more realistic
 [IMPROVED] Turrets adjust for velocity more accurately
 [IMPROVED] Auto targeting for ship turrets discard targets outside of firing cone

Version 1.1.9 – 21 September 2020

[NEW] Sample Change Camera View script
 [NEW] Ship Beam weapons
 [NEW] Destruct Module in technical preview
 [NEW] Radar - Ship Damage Regions can be tracked
 [NEW] Auto Targeting Module - include Ship Damage Regions in possible targets
 [NEW] Ship Display Module - customisable gauges with API
 [NEW] Player Input Module - call your own code with Custom Player Inputs
 [NEW] Ship Camera Module - camera shake in technical preview
 [NEW] Surface Turret Module - editor runtime debugging option
 [NEW] SampleShowShipMetrics script demonstrates creating gauges in code with API
 [FIXED] Ship Control Module - fixed weapon gizmos are incorrect when ship is rotated
 [FIXED] Radar - NullReferenceException when ship to follow is destroyed and not respawned
 [FIXED] Auto Targeting Module - Display Targets should not be shown if the HUD is not shown
 [FIXED] Auto Targeting Module - Targets may not be included when Squadrons to Include is set
 [FIXED] Player Input Module - NullReference on undo for Rewired Custom Inputs
 [FIXED] Ship Display Module - ArgumentException when adding gauges in code
 [IMPROVED] Ship Damage Regions - (destruction) effects can optionally follow a moving ship
 [IMPROVED] Radar is out of technical preview
 [IMPROVED] Radar - change the mini-map canvas sort order

Version 1.2.0 – 2 October 2020

[NEW] Full playable SSC Tech2 demo game
 [FIXED] Player Input Module - CustomPlayerInput does not contain a definition for actionIDsPositive
 [IMPROVED] Change display or screen output with HUD, Ship Camera Module, and Radar via API.
 [IMPROVED] Celestials (background stars) script supports Universal Render Pipeline

Version 1.2.1 – 10 November 2020

[NEW] Shield Recharge - damage regions have optional shield recharge rate and delay
 [NEW] Ship Control Module - Input Control to achieve 2.5D flight behaviour
 [NEW] SampleFlyToPosition - AI sample script
 [NEW] Ship AI Input Module - Button to estimate the radius of a ship for obstacle avoidance
 [NEW] Ship Camera Module - can stop camera rendering and moving at initialisation time
 [NEW] Ship Camera Module - Clip Objects is in Technical Preview
 [FIXED] PlayerInputModule - Custom Player Input Negative Key for Direct Keyboard not working
 [FIXED] ShipDisplayModule - Gauge Keep Aspect Ratio is incorrect after (I)nserting within the editor
 [IMPROVED] ShipDocking supports docking manoeuvres with and without entry or exit paths
 [IMPROVED] ShipDocking - option to detect collisions when docked
 [IMPROVED] PlayerInputModule - can disable input on initialise

Version 1.2.2 – 11 January 2021

[NEW] Auto opening and locking doors (useful for Entry/Exit scenarios)
 [NEW] Instructions to convert Tech Demo scene to (new) Unity Input System
 [NEW] Ship Docking Station - export and import to JSON options for ship docking points
 [NEW] Ship Docking - Catapult undocking option
 [NEW] Docking Demo scene as seen in docking tutorial video
 [FIXED] Player Input Module - correctly handle deleted Action Categories for Rewired integration
 [FIXED] The name 'Celestials' does not exist in the current context when not importing the Demos
 [FIXED] SSC Manager - EndLayoutGroup when exporting a Path as JSON.
 [IMPROVED] Ship Display Module (HUD) is out of technical preview

Version 1.2.3 – 8 February 2021

[NEW] Ship Control Module - Turret Beam weapons [Technical Preview]
 [NEW] Surface Turret Module - Beam weapon [Technical Preview]
 [NEW] Auto Targeting Module - Can control Turret Beam weapons [Technical Preview]
 [NEW] Ship Camera Module - Dynamic Target Offset damping [Technical Preview]
 [NEW] Paths - Insert Before in SSCManager Editor option (callable from API)
 [FIXED] PlayerInputModule API - DisableAIMode() method name spelling
 [IMPROVED] Ship Camera Module - Camera Shake now out of Technical Preview
 [IMPROVED] Ship Docking - undocking AI Ships become idle at end of exit path

Version 1.2.4 – 29 March 2021

[NEW] ShipCameraModule - Top-down setup options for 2.5D games
 [NEW] ShipDocking - can transition to docking while undocking and vis versa
 [NEW] ShipDocking - target take-off and landing durations
 [NEW] ProjectileModule - can override behaviour in own inherited class
 [NEW] SampleProjectileModule script
 [NEW] SampleCreatePath script
 [NEW] Moving Platforms for vehicle, equipment, or character elevators
 [NEW] Demo elevator sounds
 [NEW] SSCDoorAnimator supports opening and closing audio clips
 [NEW] ShipDisplayModule - ToggleHUD API method
 [NEW] ShipControlModule - editor runtime debugger for thrusters, health, and weapons
 [NEW] ShipInputModule - editor option to enable only Custom Player Inputs on initialise
 [NEW] Turret weapons - option to auto-park after interval
 [FIXED] Turret with Check Line of Sight - MissingReferenceException
 [IMPROVED] Paths - auto-adjust control points on first location when second added
 [IMPROVED] AutoTargetingModule - option to show first 10 radar query results
 [IMPROVED] DOTS compatibility for projectiles with Unity 2020.3 LTS

Version 1.2.5 – 14 June 2021

[NEW] SampleStickToGroundChange script
 [NEW] SSC Moving Platforms support kinematic rigidbodies
 [FIXED] Ship Camera Module - jitter in Camera Rotation Mode "Follow Velocity"
 [FIXED] Thruster FX and audio do not stop when Max Thrust is set to 0 at runtime
 [IMPROVED] SSC Door Proximity - uses first trigger collider
 [IMPROVED] ShipControlModule - braking algorithm

Version 1.2.6 – 28 July 2021

[NEW] Destructible Object Module - for use with non-ship objects
 [NEW] General purpose Proximity component
 [NEW] Limit Pitch/Roll - Avoid Ground Surface [Technical Preview]
 [NEW] Light Strobe utility
 [NEW] ShipDisplayModule - HUD flicker effect with API
 [NEW] Door (button) Control component
 [FIXED] IndexOutOfRangeException with Stick To Ground Surface and Smoothed Normal calc.
 [FIXED] Ship Camera Module - Target Offset Align button does not consider rotation
 [FIXED] PlayerInputModuleEditor - The name inputAxisModeContent does not exist
 [IMPROVED] All components have component-add menus
 [IMPROVED] DestructModule is out of Technical Preview
 [IMPROVED] Display Module - reticle is more accurate when Lock Reticle to Cursor is enabled
 [IMPROVED] Ships can become visible to radar in same frame as they are enabled via the API

Version 1.2.7 – 2 September 2021

- [NEW] Damage regions or whole ships can be made Invincible to damage
- [NEW] Override shipInput from ShipAllInputModule in your own game code
- [NEW] SampleInputAIOverride sample script
- [NEW] Thruster can optionally burn fuel at different rates
- [FIXED] Regression bug on ships not undocking
- [IMPROVED] Use input sensitivity and gravity in your own custom input code
- [IMPROVED] Door (button) Control component includes static API methods
- [IMPROVED] Docking manoeuvres
- [IMPROVED] Braking on approach to a target

Version 1.2.8 – 14 October 2021 (Last version to support U2018.4 LTS)

- [NEW] Thruster heat management
- [NEW] Central thruster fuel or individual fuel levels
- [NEW] Player Input Module - Unity XR [Technical Preview]
- [NEW] Support for Oculus Quest 2 [Technical Preview]
- [NEW] Object Targetable component for adding single objects to radar
- [NEW] ShipDisplayModule - gauges can have adjustable medium colour range
- [NEW] Track the number of times a ship is respawned
- [NEW] HUD scrollable heading or compass display with API
- [NEW] HUD numeric gauges with or without labels
- [NEW] Sample Tele-Port World script
- [IMPROVED] DoorControl - check for unlocked before changing button status
- [IMPROVED] Radar mini-map overlay and background are changeable at runtime in the editor
- [IMPROVED] Ship Display Module - Cursor is hidden when Lock Reticle to Cursor is on
- [IMPROVED] Ship Display Module - optional Fill Method None for numeric gauges
- [IMPROVED] Compatibility with Unity 2021.2

Version 1.2.9 – 5 November 2021

- [NEW] ShipDisplayModule - HUD attitude overlay with pitch ladder
- [FIXED] 'CustomPlayerInput' does not contain a definition for 'xrPositiveInputActionMapId'
- [FIXED] ShipDisplayModule - Heading scrolls in the wrong direction
- [FIXED] ShipDisplayModule - selecting a HUD prefab asset can create orphaned panels in the scene
- [IMPROVED] Moving Platforms include optional arrival and departure events
- [IMPROVED] Path editing - Option to Modify position Y-axis on selected Locations
- [IMPROVED] Path editing - context menu to snap Locations to mesh below

Version 1.3.0 – 3 January 2022

- [NEW] TechDemo3
- [NEW] AI targeting accuracy
- [NEW] Opening and closing events for SSCDoorAnimator
- [NEW] Demo celestials component can add planets to the background of stars
- [NEW] ShipDisplayModule - Targets debugging in editor at runtime
- [NEW] Option for weapons to be used when ship movement is disabled
- [FIXED] Docked ships can drift after being disabled and re-enabled via EnableShip APIs
- [FIXED] Destruct Modules with a Despawn condition of Time do not despawn correctly when Pooled
- [IMPROVED] Moving Platforms can smoothly stop and start
- [IMPROVED] DestructibleObjectModule is out of Technical Preview
- [IMPROVED] SSCStrobe light now works in HDRP
- [IMPROVED] AI steering behaviours

Version 1.3.1 – 21 January 2022

- [NEW] ProjectileModule - Collision Layer Mask

[FIXED] IndexOutOfRangeException after deleting local damage regions
 [IMPROVED] PlayerInputModule - add RemoveListeners() API
 [IMPROVED] UnityXR camera follows both HMD position and rotation

Version 1.3.2 – 15 March 2022

[NEW] Stability Flight Assist
 [NEW] Physics Model - Disable Drag Moments
 [NEW] Aero - Drag Moment Multipliers for pitch, yaw, and roll
 [NEW] Ship Display Module - Export-Import Gauges and Messages as JSON files
 [NEW] Ship AI Input Module - Raycast offset for colliders that do not overlap ship centre
 [FIXED] Ship Display Module - HUD3 prefab does not display heading or attitude
 [FIXED] SSC Door Animator - ToggleDoors does not play audio clips
 [FIXED] Can place multiple Ship AI components on one ship
 [IMPROVED] ThrusterEffects - virtual public methods
 [IMPROVED] SSC Door Animator - Get parameters is more reliable at design time

Version 1.3.3 – 25 May 2022

[NEW] SSCManager - Reverse the direction of Paths
 [NEW] Ship AI - optional AIState change notifications
 [NEW] ShipDockingStation - On Pre Undock and On Post Docked events
 [NEW] ShipDocking - Undocking Delay and Auto Undock Time options
 [NEW] ShipControlModule - Start-up and Shutdown Thruster Systems
 [NEW] ShipControlModule - Per Thruster throttle option
 [NEW] Ship weapon out of ammo callback
 [NEW] Floating Point Demo scene
 [FIXED] Ship AI may not move toward target when target ship gravity is 0
 [FIXED] Celestials camera may not rotate in step with main camera
 [FIXED] Celestials may raise an error if more than 1000 stars are used
 [FIXED] Ship AI - callbackCustomIdleBehaviour has the wrong type
 [FIXED] Ship AI - callbackCustomSeekMovingArrivalBehaviour has the wrong type
 [FIXED] Ship AI - callbackCustomDockBehaviour has the wrong type
 [FIXED] SSCRadar.GetRadarResults returns false
 [IMPROVED] PlayerInputModule support to invert axis in Unity Input System
 [IMPROVED] SSCDoorAnimator - Add OpenDoorsAll and CloseDoorsAll methods
 [IMPROVED] SSCRadar - Debug mode now shows Display Forwards in Euler angles
 [IMPROVED] Muzzle FX can be parented to weapon fire points

Version 1.3.4 – 21 June 2022

[NEW] Weapon heat management
 [NEW] ShipDisplayModule - ToggleDisplayReticle API
 [IMPROVED] Ship Hit callback now includes sourceShipId

Version 1.3.5 – 29 August 2022

[NEW] ProjectileModule - Shield hit effects
 [NEW] DamageRegion - IsHit API
 [NEW] Ship - HasActiveShield API
 [NEW] Ship - Attach[Detach]Collider(s) APIs
 [NEW] ShipControlModule - Ship Input Debugging in editor at runtime
 [NEW] ShipControlModule - CallbackOnCollision for custom collision handling
 [FIXED] Handles.FreeMoveHandle change in Unity 2022.1
 [IMPROVED] Projectiles support DOTs Entities 0.51 (URP and HDRP only)
 [IMPROVED] Integration with Sticky3D Controller VR flight-stick and levers

Version 1.3.6 – 17 October 2022

[NEW] EnableShip API configurable from inspector event or callback method
 [NEW] PlayerInputModule - Enable or Disable XR Camera and Hand APIs
 [NEW] ShipDocking - Events configurable in the inspector and supporting APIs
 [NEW] Thrusters - optional FX when stationary or no thruster input
 [NEW] ShakeCameraDelayed and ShakeCamera API callable from inspector events
 [NEW] SetThrusterThrottle API for setting individual thrusters
 [NEW] SetMaxThrustNewtons API for fine-grained control with ultra-light weight ships
 [NEW] Ship callbackOnWeaponFired delegate
 [FIXED] Thrusters may have Throttle > 0 when Thruster Systems have not started
 [IMPROVED] Disable XR camera and hands when Enable on Initialise is disabled
 [IMPROVED] Shutdown[Startup]ThrusterSystems APIs callable from inspector events
 [IMPROVED] Path editing - add or subtract y-axis position for multiple selected Locations

Version 1.3.7 – 23 February 2023

[NEW] Ship Warp Module with API (in technical preview)
 [NEW] Camera Module - Zoom with API
 [NEW] DemoSimpleGroupLOD script - render culling for groups of non-LOD objects
 [NEW] ShipDisplayModule (HUD) - fade messages in and out
 [NEW] Thrusters - option to adjust minimum effects rate by throttle
 [FIXED] DisableShipMovement and EnableShipMovement do not check if ship is initialised
 [FIXED] Thrusters Started at runtime in the editor
 [FIXED] Thruster Systems may not come online in some scenarios
 [FIXED] Default font in Unity 2022.2+
 [FIXED] Muzzle FX not returned to pool when damage region of weapon destroyed
 [FIXED] Tech Demo 2 mission end menu may not appear
 [IMPROVED] SSCManager - Context menu - start or stop editing a path in scene view
 [IMPROVED] ShipDockingStation - Events have their own tab in the editor
 [IMPROVED] Enhanced path following for Physics Model ships
 [IMPROVED] Celestials - options to support jump gate scenarios for built-in and URP
 [IMPROVED] Updated link to tutorials
 [IMPROVED] Updated documentation links

Version 1.3.8 - 03 May 2023

[NEW] Brake Assist optional X and Y axis
 [NEW] SSCManager - support for additive scenes
 [NEW] Sample Component for Arcade Variable Flight Acceleration
 [FIXED] Input Control 2.5D may not return to original plane when bumped off course
 [IMPROVED] ShipControlModule - debug shows x and y local velocities

Version 1.3.9 – 11 June 2023

[NEW] AddWeapon API
 [FIXED] May not respawn at correct location in U2022+ with Vsync or targetFrameRate set
 [IMPROVED] Sticky3D Controller first-person integration with TechDemo3
 [IMPROVED] Compatibility with Unity 2022.3 and 2023.1

Version 1.4.0 – 03 August 2023

[NEW] Ship Display Module – Fade in or out options with API
 [NEW] DemoSSCSequenceRenderers component
 [NEW] Ship Proximity component
 [NEW] Ship AI Input Module - ability to enable or disable main update loop
 [FIXED] SSCManager may not be created in correct scene when using additive scenes
 [FIXED] Copied style is null. Using StyleNotFound instead (U2022.3 only)
 [FIXED] ShipDocking - on undock event does not fire if there is no exit path
 [IMPROVED] Ship Camera Module - Check Update Type if Lock To Pos or Rot

Trade Marks

“Unity” is a trade mark of Unity Technologies and is in no way associated with SCSM Pty Ltd.

Other names or brands are trademarks of their respective owners.